IL MANUALE DELL'HARDWARE DELL'AMIGA

IHT GRUPPO EDITORIALE

COMMODORE-AMIGA, INC.

LIBRERIA DI RIFERIMENTO TECNICO ALL'AMIGA

IL MANUALE DELL'HARDWARE DELL'AMIGA

IL MANUALE DELL'HARDWARE DELL'AMIGA

PRIMA EDIZIONE

COMMODORE-AMIGA, INC.

PREFAZIONE

La serie di manuali di riferimento tecnico a cui appartiene questo volume è la guida ufficiale per la programmazione dell'Amiga. Si tratta di un'edizione riveduta, aggiornata alla versione 1.3 del sistema operativo e ai modelli più recenti dell'Amiga.

Questo manuale contiene informazioni sui dispositivi hardware che governano la grafica, il sonoro e il "dialogo" tra computer e periferiche. Contiene moltissimi esempi in linguaggio Assembly, che mostrano come controllare direttamente l'hardware e la grafica dell'Amiga.

Il Manuale dell'Hardware dell'Amiga si rivolge principalmente alle seguenti categorie di persone:

- Programmatori in linguaggio Assembly che hanno bisogno di interagire con la macchina in maniera diretta.
- Progettisti che intendono creare nuove periferiche per l'Amiga.
- Chiunque sia interessato a scoprire come funziona l'hardware dell'Amiga.

Ecco una breve descrizione del contenuto del manuale:

Capitolo 1, *Introduzione.* Una panoramica sull'hardware e un riassunto delle capacità grafiche e musicali dell'Amiga.

Capitolo 2, *Hardware del Copper.* Come utilizzare il coprocessore Copper per controllare il sistema audio e video; come modificare le caratteristiche di un'immagine mentre viene visualizzata e come controllare i registri tra un quadro video e il successivo.

Capitolo 3, *Hardware dei playfield.* Come creare, visualizzare e far muovere sullo schermo i playfield, elementi fondamentali nelle applicazioni video. Come far visualizzare alla macchina immagini di diversa risoluzione e numero di colori.

Capitolo 4, *Hardware degli sprite.* Come utilizzare gli otto canali di accesso diretto alla memoria (DMA) degli sprite per produrre oggetti in movimento sullo schermo; come organizzare le loro strutture dati, come riutilizzare i canali più volte nello stesso quadro video.

Capitolo 5, *Hardware audio.* Panoramica sulla digitalizzazione del suono; come produrre suoni di buona qualità, semplici e complessi, e come modulare un suono.

Capitolo 6, *Hardware del Blitter.* Come usare il canale DMA del Blitter per creare effetti d'animazione e per tracciare linee all'interno dei playfield.

Capitolo 7, *Hardware di controllo.* Come usare i registri di controllo per definire le priorità degli oggetti grafici, per rilevare le collisioni, per controllare i canali DMA e gli interrupt.

Capitolo 8, *Hardware d'interfaccia.* Come l'Amiga dialoga con il mondo esterno attraverso le porte di controllo, la tastiera, i jack audio, i connettori video, le interfacce seriale e parallela; informazioni sul controller dei disk drive e sugli slot d'espansione.

Appendici. Lista dei registri in ordine alfabetico e d'indirizzo, con il significato di ogni bit; mappa della memoria del sistema; descrizione dei connettori interni ed esterni; specifiche per l'interfacciamento con le periferiche e con la tastiera.

Glossario. Glossario dei termini più significativi.

Suggeriamo di utilizzare questo libro in modo diverso a seconda della propria familiarità con il sistema.

- Se si tratta del primo contatto con l'Amiga, consigliamo di leggere il Capitolo 1, che offre una panoramica di tutte le caratteristiche grafiche e musicali del sistema, e spiega come sfruttarle agendo sull'hardware.

- Se si possiede già una certa familiarità con il sistema e si vogliono approfondire le proprie conoscenze, i capitoli dal 2 all'8 spiegano come i registri hardware agiscono sulle funzioni del sistema, e propongono alcuni esempi pratici.

- Per i programmatori più esperti, le appendici forniscono un conciso riepilogo dell'intera serie di registri e del significato di ogni bit. Una volta che si sarà raggiunta una buona familiarità con gli effetti di ogni bit, questa diventerà probabilmente la parte più consultata del volume.

L'altro manuale della serie s'intitola *Il Manuale del ROM Kernel dell'Amiga* ed è diviso in due tomi: *Librerie e dispositivi*, che contiene spiegazioni dettagliate sull'uso di ogni libreria e di ogni dispositivo dell'Amiga, e *File Include e Autodoc*, che elenca in ordine alfabetico tutti gli autodoc di ogni funzione, dei file include di sistema, e le specifiche di formato dei file IFF.

Commodore Amiga Technical Support (CATS)

La Commodore mantiene un gruppo di supporto tecnico per aiutare i progettisti a raggiungere i loro obiettivi. L'aiuto è disponibile sia per gli indipendenti sia per le grandi società. Chiunque

sia interessato alle ultime novità, a informazioni sui cambiamenti hardware o software e a consigli sulla programmazione, può abbonarsi alla pubblicazione di supporto *AmigaMail*.

Per informazioni sul programma Commodore, scrivere a:

CATS-Information
1200 West Wilson Drive
West Chester, PA 19380-4231
USA

Errori nel manuale

In un complesso manuale tecnico vengono spesso scoperti degli errori dopo la pubblicazione. Gli errori individuati vengono corretti nell'edizione successiva, ed eventuali aggiornamenti compaiono sulla pubblicazione di supporto tecnico *AmigaMail*.

Gli errori si possono segnalare alla Commodore tramite lettera o via modem. Le informazioni devono essere chiare, complete e concise. Devono inoltre includere un numero telefonico ed esaurienti indicazioni sul modo di verificare l'errore.

Amiga Software Engineering Group
ATTN: BUG REPORTS
Commodore Business Machines
1200 Wilson Drive
West Chester, PA 19380-4231
USA

BIX: afinkel
USENET: bugs@commodore.COM oppure uunet!cbmvax!bugs
oppure suggestions@commodore.COM

Se non volete inviare le segnalazioni di errori negli Stati Uniti, potete comunque inviarle in Italia all'indirizzo riportato qui di seguito. Se inviate tali segnalazioni negli Stati Uniti, siete comunque pregati di inviarle anche a:

IHT Gruppo Editoriale
Divisione Libri
Redazione del Manuale dell'Hardware dell'Amiga
Via Monte Napoleone, 9
20121 Milano

SOMMARIO

Elenco delle figure

Elenco delle tavole

1 INTRODUZIONE

La famiglia di computer Amiga è costituita da diversi modelli, ognuno dei quali è stato disegnato con lo stesso obiettivo: fornire all'utente un elaboratore a basso costo con caratteristiche riscontrabili solo su modelli di prezzo superiore. L'Amiga raggiunge questo scopo tramite appositi chip costruiti in modo da fornire una resa grafica e sonora di altissimo livello.

Sono tre i modelli principali della linea Amiga: A500, A1000 e A2000. Sebbene siano diversi fra loro per dimensioni e prezzo, hanno un nucleo hardware comune che li rende "software compatibili" l'uno con l'altro. Questo capitolo descrive i componenti hardware dell'Amiga e dà un breve sommario delle sue capacità grafiche e sonore.

I componenti dell'Amiga

Questi sono i componenti hardware dell'Amiga:

- Microprocessore Motorola MC68000 a 16/32 bit. L'Amiga prevede anche l'installazione opzionale di un 68010, 68020 o 68030.
- 512K di RAM interna, espandibile a 1 MB su A500 e A2000.
- 256K di ROM, contenente un sistema operativo multitasking in tempo reale, con gestione del suono, della grafica e dell'animazione.
- Drive interno per dischi da 3,5"
- Porta di espansione per collegare fino a tre disk drive a doppia faccia supplementari, sia da 3,5" che da 5,25"
- Porta seriale RS-232-C completamente programmabile.

- Porta parallela completamente programmabile.

- Mouse optomeccanico a due pulsanti.

- Due porte di controllo riconfigurabili (per mouse, joystick, penne ottiche, paddle o controlli di altro tipo).

- Una tastiera professionale separata con tastierina numerica, dieci tasti funzione e tasti di controllo cursore. Sono previste varie tastiere internazionali.

- Porte di output per video videocomposito e RGB analogico o digitale.

- Porte di output destra e sinistra, per segnali audio provenienti da quattro canali.

- Caratteristiche di espandibilità che permettono l'aggiunta di RAM, hard disk, disk drive, periferiche e coprocessori.

IL MICROPROCESSORE MC68000 E I CHIP CUSTOM DELL'AMIGA

Il Motorola 68000 è un microprocessore a 16/32 bit. Il clock di sistema per gli Amiga in standard PAL è di 7,09379 MHz (7,15909 MHz in NTSC). Queste velocità possono cambiare con l'uso di una sorgente di clock esterna, come un genlock. Il 68000 ha uno spazio d'indirizzamento di 16 MB. Nell'Amiga il 68000 può indirizzare più di 8 MB di memoria RAM.

Oltre al 68000, l'Amiga contiene elementi specializzati noti come "chip custom", che aumentano moltissimo le prestazioni del sistema. Il termine "chip custom" si riferisce a tre circuiti integrati disegnati appositamente per il computer Amiga. Ognuno di questi tre chip (chiamati Agnus, Paula e Denise) contiene la logica necessaria al controllo di compiti specifici, quali l'audio, il video, l'accesso diretto alla memoria (DMA) o la grafica.

Ecco alcune delle funzioni dei chip custom:

- Grafica ad alta risoluzione generata tramite bitplane, in standard PAL o NTSC.

  - Nel sistema NTSC l'Amiga può generare display 320 x 200 non interlace o 320 x 400 interlace con un massimo di 32 colori, e 640 x 200 non interlace o 640 x 400 interlace con un massimo di 16 colori.

  - Nel sistema PAL l'Amiga può generare display 320 x 256 non interlace o 320 x 512 interlace con un massimo di 32 colori, e 640 x 256 non interlace o 640 x 512 interlace con un massimo di 16 colori.

  Altri modi video permettono la visualizzazione di 64 o 4096 colori contemporaneamente (Extra Half-Brite e Hold and Modify) o permettono la creazione di display di dimensioni maggiori (overscan).

- Un coprocessore video che può modificare gran parte dei registri hardware in sincronia con la posizione del pennello elettronico. Ciò permette effetti speciali quali il cambio della palette a metà dello schermo, la divisione dello schermo in settori orizzontali aventi diversa risoluzione e numero di colori, la generazione di interrupt del 68000 e altro ancora. Il coprocessore può agire diverse volte durante ogni quadro video, tra una linea di scansione e l'altra, all'inizio o durante l'intervallo di vertical blanking. Il

coprocessore può agire su gran parte dei registri contenuti nei chip custom, liberando il 68000 da molti compiti di carattere generale.

- 32 registri di colore, ognuno dei quali contiene un numero a 12 bit costituito da quattro bit d'intensità del rosso, quattro del verde e quattro del blu. Ciò permette una palette di 4096 colori per ogni registro.

- Otto sprite riutilizzabili larghi 16 bit con una scelta di 15 colori per pixel (quando gli sprite sono accoppiati). Uno sprite è un oggetto grafico mobile la cui immagine è completamente indipendente dallo sfondo (detto playfield); gli sprite possono essere visualizzati sopra o sotto questo sfondo. Uno sprite ha una larghezza di 16 pixel e un'altezza arbitraria. Dopo aver prodotto l'ultima linea di uno sprite sullo schermo, il canale DMA associato può essere riutilizzato per produrre un'altra immagine in un altro punto dello schermo (separata da almeno una linea di scansione). Riutilizzando i canali DMA in questo modo, si possono produrre molti piccoli sprite.

- Priorità tra gli oggetti video controllabile dinamicamente, con rilevazione delle collisioni. Ciò significa che il sistema può controllare la priorità tra gli sprite e i bitplane di sfondo (playfield). In ogni momento si può stabilire quali oggetti sono "sopra" lo sfondo e quali sotto. L'hardware rileva inoltre le eventuali collisioni tra gli oggetti video.

- Il Blitter, un dispositivo in grado di muovere i dati ad alta velocità e utilizzabile per le animazioni. Il Blitter è stato disegnato in maniera da poter raccogliere dati da tre diverse sorgenti, combinarli tra loro in un massimo di 256 modi e trasferirli in un'altra area di memoria. Questa è una delle situazioni in cui può accadere che il 68000 sia costretto a cedere cicli di memoria a favore di un canale DMA che può svolgere il suo stesso lavoro con maggiore efficienza. Il Blitter sfrutta una speciale tecnica di tracciamento che gli consente di disegnare una linea alla velocità di circa un milione di punti al secondo; è inoltre particolarmente efficiente nel riempimento delle aree.

- Quattro canali audio digitali con volume e frequenza di campionamento programmabili separatamente. I canali audio ricevono i dati da appositi canali DMA. Una volta inizializzato, ogni canale può riprodurre automaticamente una specifica forma d'onda senza ulteriore intervento del microprocessore. A ciascuna uscita stereo arrivano due canali. I canali audio possono essere collegati tra loro per ottenere effetti di modulazione di frequenza o di ampiezza, o di entrambe simultaneamente.

- Lettura e scrittura da disco controllate tramite DMA, a blocchi di una traccia per volta. Ciò significa che il drive interno può leggere oltre 5600 byte di dati in una sola rotazione del disco (11 settori di 512 byte ciascuno).

La memoria interna condivisa dai chip custom e dal 68000 viene anche detta "memoria chip" I chip custom originali dell'Amiga erano in grado di accedere ai primi 512K della memoria. Il nuovo Super Agnus, disponibile dalla seconda metà del 1989 su A500 e A2000, ha reso accessibile all'hardware grafico e sonoro un intero megabyte di memoria. Sull'Amiga 1000, invece, la memoria chip è rimasta limitata ai primi 512K.

I chip custom e il 68000 accedono alla memoria in maniera alternata. Dal momento che, per procedere alla velocità massima, il 68000 ha necessità di accedere al bus di memoria solo una volta ogni due cicli di clock, per il resto del tempo il bus rimane libero per altre attività. I chip custom utilizzano il bus di memoria durante questi cicli, permettendo al 68000 di funzionare alla

massima velocità per la maggior parte del tempo. Diciamo "per la maggior parte del tempo", perché vi sono alcune occasioni in cui l'hardware può rubare cicli di memoria al 68000. In particolare possono farlo il coprocessore Copper e il Blitter, che sono in grado di eseguire alcuni compiti più efficacemente del 68000. Il sistema di canali DMA è stato progettato per ottenere il massimo rendimento possibile: il lavoro viene sempre svolto dall'elemento hardware più efficiente. In ogni caso, quando è necessario l'uso di questi cicli supplementari da parte dei chip custom, il 68000 viene bloccato soltanto nell'accesso alla memoria chip. Quando utilizza la ROM o la RAM esterna, il 68000 lavora sempre alla massima velocità.

Un'altra caratteristica basilare dell'hardware dell'Amiga è la capacità di controllare in maniera dinamica quale parte della memoria chip dev'essere usata per il display, l'audio, gli sprite. La memoria dell'Amiga non assegna aree specifiche a queste caratteristiche. Al contrario, il sistema permette che i bitplane, i dati degli sprite, le istruzioni del Copper e i dati dei canali audio siano collocati in qualunque posizione all'interno della memoria chip.

La memoria chip è anche quella accessibile al Blitter. Questo significa, per esempio, che un certo programma può memorizzare diverse immagini parziali in punti casuali della memoria chip e utilizzarle in seguito per effetti d'animazione rimpiazzando rapidamente quelle sullo schermo (badando a salvare e ripristinare ogni volta lo sfondo). Il firmware contenuto in ROM rende disponibili, inoltre, funzioni per il controllo dei playfield e degli oggetti contenuti al loro interno.

INTERFACCIAMENTO CON TELECAMERA E VIDEOREGISTRATORE

Oltre ai connettori per i monitor RGB e videocompositi, l'Amiga può accogliere anche un'interfaccia per videocamera o videoregistratore. Il sistema è capace di sincronizzarsi con una sorgente video esterna e di utilizzarla per rimpiazzare il colore di sfondo delle immagini. Ciò facilita lo sviluppo d'immagini video integrate con grafica generata dal calcolatore. Con lo stesso sistema il computer può accettare anche input da videodischi laser.

PERIFERICHE

Come memoria di massa di base, l'Amiga include un drive da 3,5". I dischi contengono 80 tracce, sono a doppia faccia, con 11 settori per traccia e 512 byte per settore (più di 900 mila byte per disco). Il controller può leggere e scrivere dischi da 320/360K del tipo IBM PC (MS-DOS) sia da 3,5" che da 5,25", e dischi da 640/720K del tipo IBM PC (MS-DOS) da 3,5". Drive aggiuntivi da 3,5" o 5,25" possono essere aggiunti al sistema tramite l'apposito connettore di espansione.

La circuiteria di alcune periferiche è contenuta nel chip Paula. Altri chip gestiscono segnali non assegnati specificamente ai chip custom come il controllo del modem, la rilevazione dello stato del disk drive, del suo motore e della testina di lettura e scrittura, delle interfacce parallela e seriale e della tastiera.

L'Amiga possiede un'interfaccia RS-232-C seriale standard per l'uso con dispositivi seriali esterni di input/output.

Il sistema di base comprende inoltre una tastiera con dieci tasti funzione, normale tastiera alfanumerica, tasti cursore e tastierina numerica. Per ottenere la massima flessibilità, vengono rilevati sia la pressione sia il rilascio dei tasti. Sono previste anche diverse tastiere internazionali. Alle due porte d'ingresso disponibili possono inoltre essere collegate altre unità di controllo: entrambe le porte possono accettare mouse, joystick, trackball, tavolette grafiche e penne ottiche.

ESPANSIONE E ADATTABILITÀ DEL SISTEMA

Si possono facilmente collegare nuovi dispositivi a tutti i modelli dell'Amiga. Questi vengono automaticamente riconosciuti e utilizzati dal software sistema tramite una precisa procedura di collegamento denominata AUTOCONFIG.

Sull'Amiga 500 e sull'Amiga 1000, i dispositivi esterni (comprese le espansioni di memoria) possono essere collegati tramite il connettore di espansione a 86 pin. I disk drive supplementari vengono collegati al connettore sul retro della macchina.

L'Amiga 2000 ha le stesse caratteristiche essenziali degli altri due modelli, ma offre parecchie possibilità in più. Il connettore a 86 pin non è accessibile dall'esterno, ma sono presenti 7 slot interni che permettono di aggiungere direttamente all'interno della macchina molti tipi di schede. Può trattarsi di coprocessori matematici, schede di memoria, controller per hard disk, porte video e di I/O. Vi è anche lo spazio per il montaggio interno di floppy o hard disk. L'A2000 permette inoltre il montaggio di una speciale scheda Bridgeboard: si tratta di un completo PC IBM su scheda, che permette all'Amiga di mandare in esecuzione software MS-DOS, utilizzando simultaneamente il software Amiga originale.

A proposito degli esempi

Gli esempi contenuti in questo libro riguardano la manipolazione diretta dell'hardware. Tuttavia, come regola generale, non è ammissibile accedere direttamente all'hardware a meno che il programma in questione non abbia il controllo totale della macchina o non abbia ottenuto l'accesso a particolari elementi hardware tramite apposite chiamate al sistema operativo.

Quasi tutto l'hardware discusso in questo manuale, e in particolare il Blitter, il Copper, i playfield, gli sprite, i CIA, il dispositivo di controllo dei dischi (TrackDisk) e l'hardware di controllo del sistema, è in uso esclusivo della macchina o comunque sotto la sua supervisione. Altri componenti dell'hardware invece, come i canali audio, la porta seriale o quella parallela, possono essere controllati dalle applicazioni.

Prima di agire su qualunque elemento hardware, un programma deve prima averne ottenuto l'accesso esclusivo dalla libreria, dispositivo o risorsa che ne controlla il possesso. Le funzioni del sistema operativo al riguardo sono molte e la loro descrizione non rientra negli scopi di questo manuale. Queste funzioni, in genere, fanno parte della libreria, dispositivo o risorsa che controlla quel particolare elemento hardware all'interno del sistema multitasking. La lista che segue può fornire un aiuto nella ricerca delle appropriate funzioni o meccanismi di accesso all'hardware discusso in questo manuale.

Copper, Playfield, Sprite, Blitter - graphics.library.
Audio - audio.device.
TrackDisk - trackdisk.device, disk.resource.
Porta seriale - serial.device, misc.resource.
Porta parallela - parallel.device, cia.resource, misc.resource.
Porte di controllo - input.device, gameport.device, potgo.resource.
Tastiera - input.device, keyboard.device.
Controllo del sistema - graphics.library, exec.library (interrupt).

La maggior parte degli esempi utilizza per i nomi dei registri il file hw_examples.i (si veda l'Appendice J). Il file hw_examples.i usa il file include hardware/custom.i per definire le strutture e l'indirizzamento relativo. I valori contenuti in questi file sono offset dalla base dello spazio d'indirizzamento dei registri. In genere questo valore di base viene definito come

_custom e il riferimento viene risolto dal linker tramite la libreria amiga.lib. Lo stesso dicasi per _ciaa e _ciab.

La procedura più usata consiste nel caricare in un registro l'indirizzo di base e accedere quindi al registro desiderato tramite gli offset contenuti in hardware/custom.i e hw_examples.i.

Anche il Copper utilizza gli stessi valori di offset per accedere ai registri.

Ad esempio, in Assembly:

```
      INCLUDE "exec/types.i"
      INCLUDE "hardware/custom.i"

      XREF    _custom               ;Riferimento esterno

Start:lea     _custom,a0            ;Utilizza a0 come registro di base
      move.w  #$7FFF,intena(a0)     ;Disabilita gli interrupt
```

In C, invece si utilizzano le strutture definite nel file hardware/custom.h. Ad esempio:

```
#include "exec/types.h"
#include "hardware/custom.h"

extern struct Custom custom;

main()
{
custom.intena = 0x7FFF; /* Disabilita gli interrupt */
}
```

I file include riguardanti l'hardware dell'Amiga vengono in genere forniti insieme al compilatore o all'assembler. Sono inclusi anche nel *Manuale del ROM Kernel: file Include e Autodoc*. In genere i nomi contenuti in questi file sono simili a quelli convenzionali dei registri, con le seguenti differenze:

- I registri d'indirizzamento costituiti da due word distinte sono elencati come due registri distinti da 16 bit. I nomi si differenziano per la presenza di un suffisso "L" (low) o "H" (high) per distinguere le due parti. Nei file include invece vengono in genere trattati come un'unica long word da 32 bit che perciò non presenta distinzioni di "H" o "L".

- I registri che si ripetono in sequenze ordinate appaiono nel relativo elenco hardware con un suffisso numerico, mentre sono in genere definiti da un unico valore di base nei file include. Per esempio, i registri di colore (COLOR00, COLOR01...) vengono utilizzati tramite la label "color" definita nel file hardware/custom.i (color+0, color+2...).

- Nell'Appendice J, e per la precisione nel file hw_examples.i, si trovano alcuni esempi di corretta definizione degli offset dei registri.

Alcune avvertenze per i programmatori

L'Amiga è disponibile in molti modelli e configurazioni, ulteriormente diversificabili da una vasta gamma di periferiche e di schede opzionali. Per di più, anche l'hardware standard della macchina (come la tastiera o i disk drive) viene fornito da diverse case produttrici e nel timing o nelle prestazioni può uscire dalle specifiche previste.

Il sistema operativo dell'Amiga è stato disegnato in maniera da funzionare con tutto l'hardware che si mantiene entro le specifiche dichiarate anche con diverse configurazioni di RAM. In generale presenta una notevole compatibilità verso l'alto, ovvero verso futuri miglioramenti e aggiunte al sistema. Per garantire al massimo questa compatibilità, si consiglia ai programmatori di utilizzare le risorse hardware tramite le funzioni messe a disposizione dal sistema operativo.

Se non vi è altra soluzione che accedere direttamente all'hardware, allora è responsabilità del programmatore scrivere un codice che funzioni con tutti i modelli e tutte le configurazioni. È comunque sempre necessario richiedere al sistema il controllo dell'area su cui s'intende agire, e seguire i seguenti avvertimenti:

- Non saltare direttamente in ROM. È bene evitare di prendere esempio dai programmi che chiamano direttamente le funzioni nell'area che va da $F80000 a $FFFFFF Questi sono indirizzi che appartengono alla ROM e le funzioni in ROM cambiano posizione a ogni revisione del sistema operativo. L'unica interfaccia sempre valida con la ROM è garantita dalle librerie, dai dispositivi e dalle risorse.

- Non basare i propri programmi su strutture private del sistema. Questo include la manipolazione diretta delle liste del Copper, delle liste di memoria e dei dati interni delle librerie.

- Non basare i propri programmi sul fatto che particolari indirizzi assoluti corrispondano a determinate strutture del sistema o appartengano a memoria di un certo tipo. I moduli del sistema allocano dinamicamente, al momento dell'inizializzazione, la memoria di cui hanno bisogno. Gli indirizzi delle strutture di sistema e dei buffer cambiano in ogni versione del sistema operativo, in ogni modello e in ogni configurazione dell'Amiga, così come l'ammontare della memoria libera o le dimensioni dello stack. Si tenga presente che tutti i dati necessari ai chip custom devono trovarsi nella memoria chip. Ciò include anche le immagini grafiche (bitplane, sprite...), i dati audio, i buffer dei disk drive e le liste del Copper.

- Non servirsi in alcun modo di bit o indirizzi ancora inutilizzati nello spazio dei chip custom. Tutti i bit non definiti devono essere azzerati in scrittura e ignorati in lettura.

- Non utilizzare lo spazio che segue immediatamente quello dei chip custom. Questi, infatti, potrebbero essere estesi o aggiornati per contenere altri registri o per utilizzare i bit non ancora definiti.

- Tutti i registri dei chip custom sono a sola lettura o a sola scrittura. È bene evitare di scrivere nei primi e di leggere i secondi.

- Non leggere o scrivere in spazi d'indirizzamento non utilizzati. L'uso attuale e futuro di quelle aree è riservato alla Commodore ed è soggetto a cambiamenti.

- Se si fa uso di librerie, dispositivi o risorse di sistema, occorre seguire il protocollo stabilito. I programmatori in Assembly (e i programmatori di compilatori) devono accedere alle funzioni tramite le tavole di salto della libreria corrispondente, con argomenti costituiti da long word e con l'indirizzo base della libreria inserito nel registro A6. Si deve controllare il valore restituito in D0, e considerare perduto il precedente contenuto dei registri D0-D1/A0-A1.

- L'istruzione TAS non dev'essere mai usata. Quest'istruzione presuppone un ciclo indivisibile di lettura/scrittura, ma ciò può essere invalidato dal DMA di sistema. Si usino invece BSET e BCLR. Queste istruzioni presentano un ciclo di test e impostazione che non può essere interrotto. L'istruzione TAS è necessaria soltanto su un sistema che contiene più di una CPU. In un sistema a CPU singola, le istruzioni BSET e BCLR sono identiche a TAS, dal momento che il 68000 non interrompe la loro esecuzione. BSET e BCLR controllano i bit prima d'impostarli.

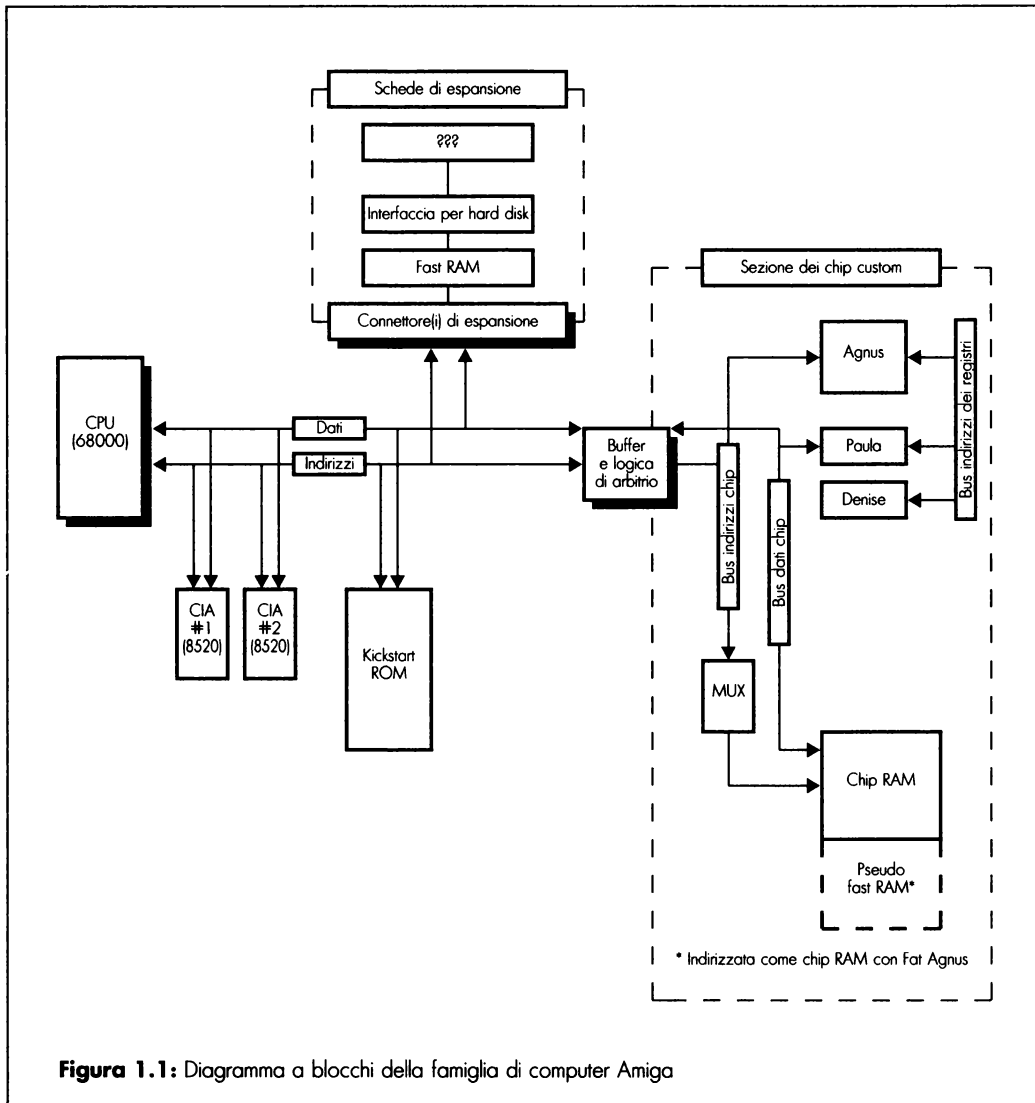
- Non utilizzare istruzioni privilegiate su nessun membro della famiglia 68000, in particolare l'istruzione MOVE SR, <ea> (che è privilegiata su 68010/20/30). Utilizzare invece la funzione GetCC() della libreria Exec, oppure le appropriate istruzioni non privilegiate:

CPU	**Modo User**	**Modo Supervisor**
68000	MOVE SR,<ea>	MOVE SR,<ea>
68010/20/30	MOVE CCR,<ea>	MOVE SR,<ea>

- Tutti gli indirizzi devono essere di 32 bit. Non inserire dati negli 8 bit superiori e non utilizzare variabili oppure operazioni con segno per gli indirizzi. Non eseguire codice sullo stack e non usare codice che si automodifichi, perché può diventare inutilizzabile a causa delle capacità di cache di alcuni microprocessori 680xx. Non utilizzare mai loop dipendenti dalla velocità della CPU o dal clock per causare ritardi nei programmi. Si consulti l'Appendice F per informazioni riguardanti l'uso dei timer dei chip 8520.

- Nell'accedere a un indirizzo strobe che risponde sia a una lettura che a una scrittura (per esempio COPJMP2), assicurarsi di usare un'istruzione MOVE.W #$00 e non CLR.W. L'istruzione CLR causa una lettura e una scrittura con il 68000 (due accessi), ma un solo accesso con il 68020 o 68030. Ciò dà quindi origine a risultati diversi con CPU diverse.

- Nel programmare a livello hardware è necessario seguire una corretta procedura d'interfacciamento. L'hardware non è sempre uguale. È sbagliato ritenere che manipolazioni a basso livello per la velocità o la protezione di un programma funzioneranno su ogni disk drive, su ogni tastiera, su ogni sistema. Ogni programma va verificato su vari sistemi, con processori diversi, diversi sistemi operativi, diverse configurazioni di memoria.

Figura 1.1: Diagramma a blocchi della famiglia di computer Amiga

2 HARDWARE DEL COPPER

Introduzione

Il Copper è un coprocessore situato in uno dei chip custom dell'Amiga. Esso ottiene le sue istruzioni tramite accesso diretto alla memoria (DMA). Il Copper può controllare praticamente l'intero sistema grafico, permettendo al 68000 di occuparsi d'altro; può anche agire direttamente sul contenuto di gran parte dei registri di controllo contenuti nei chip custom. Il Copper è utilissimo per modificare l'immagine video durante la visualizzazione e per controllare le variazioni del contenuto dei registri durante l'intervallo di vertical blanking. Tra le altre cose, controlla l'aggiornamento dei registri video, la posizione degli sprite, modifica la palette dei colori, aggiorna i canali audio e può controllare il Blitter.

Una delle caratteristiche del Copper è la capacità di attendere, tramite l'istruzione WAIT, che il pennello elettronico raggiunga una specifica posizione sullo schermo, e quindi, tramite l'istruzione MOVE, di modificare il contenuto di un determinato registro hardware. Durante l'attesa, il Copper esamina direttamente il contenuto del registro di controllo della posizione del pennello elettronico. Ciò significa che mentre attende che venga raggiunta una determinata posizione, il Copper non usa affatto il bus di memoria, e quest'ultimo rimane pertanto libero per il 68000 o altri canali DMA.

Quando la posizione desiderata viene raggiunta, il Copper ruba cicli di memoria al Blitter o al 68000 per modificare i dati del registro selezionato.

Il Copper è un processore a due cicli, e richiede l'accesso al bus soltanto durante i cicli dispari. In tal modo si previene ogni sovrapposizione con il DMA audio, i disk drive, il refresh della memoria, gli sprite e i display in bassa risoluzione, i quali utilizzano esclusivamente i cicli pari. Il Copper, perciò, richiede una priorità più alta soltanto rispetto al 68000 e al Blitter (il canale DMA che gestisce l'animazione, il tracciamento di linee e il riempimento di aree).

Come tutti gli altri canali DMA dell'Amiga, il Copper può accedere alle sue istruzioni soltanto se queste si trovano in un'area di memoria di tipo chip.

PRESENTAZIONE DEL CAPITOLO

In questo capitolo vedremo come utilizzare il set d'istruzioni del Copper per modificare i registri durante la visualizzazione di un'immagine, e per preparare i registri durante l'intervallo di vertical blanking. Vedremo inoltre come si organizzano in liste ordinate le istruzioni del Copper, come si organizzano le liste in modo interlace e come utilizzare il Copper per controllare il Blitter. Qui il Copper viene trattato da un punto di vista generale, ma per indicazioni più specifiche si vedano anche i capitoli che trattano i playfield, gli sprite, l'audio e il Blitter.

Cos'è un'istruzione del Copper?

In quanto coprocessore, il Copper aggiunge un proprio set di istruzioni a quello disponibile con il 68000. Il Copper possiede soltanto tre istruzioni, ma molto importanti. Infatti può:

- Attendere (WAIT) una specifica posizione del pennello elettronico espressa tramite coordinate x e y.
- Spostare (MOVE) un valore numerico in un registro hardware.
- Saltare (SKIP) l'istruzione successiva se il pennello elettronico ha già raggiunto una certa posizione sullo schermo.

Tutte le istruzioni del Copper consistono di due word da 16 bit in locazioni di memoria contigue. Ogni volta che il Copper legge un'istruzione, legge entrambe queste word. Le istruzioni MOVE e SKIP richiedono due cicli di memoria. Dal momento che il Copper utilizza solo i cicli dispari, ogni istruzione richiede un totale di quattro cicli completi. L'istruzione WAIT, invece, richiede tre cicli di memoria (per un totale di sei cicli completi); il ciclo in più è necessario per il "risveglio".

Sebbene il Copper possa agire direttamente soltanto sui registri hardware, indirettamente può agire anche sulla memoria tramite l'uso del Blitter. Informazioni sul controllo del Blitter tramite il Copper si trovano nei paragrafi "Il registro di controllo" e "Uso del Copper con il Blitter".

Le istruzioni MOVE e WAIT sono descritte nei paragrafi seguenti. L'istruzione SKIP viene trattata nel paragrafo "Informazioni avanzate".

L'istruzione MOVE

L'istruzione MOVE trasferisce dati dalla RAM a un registro destinazione. Il dato da trasferire è contenuto nella seconda word dell'istruzione stessa, mentre la prima word contiene l'indirizzo del registro destinazione. Questa procedura verrà esaminata in dettaglio nell'ultimo paragrafo del capitolo.

PRIMA WORD DELL'ISTRUZIONE (IR1)

Bit 0 Sempre a 0.

Bit 8-1 Indirizzo del registro destinazione (DA8-1).

Bit 15-9 Non utilizzati, vanno impostati a 0.

SECONDA WORD DELL'ISTRUZIONE (IR2)

Bit 15-0 16 bit di dati da trasferire nel registro destinazione.

Il Copper può modificare i seguenti registri:

- Qualunque registro il cui indirizzo sia $80 o maggiore.[1]
- Qualunque registro il cui indirizzo sia compreso fra $40 e $80 se il bit di pericolo del Copper è impostato a 1. Il bit di pericolo del Copper si trova nel registro COPCON, e viene descritto nel paragrafo "Il registro di controllo".
- Il Copper non può accedere ai registri il cui indirizzo è minore di $40.

Si veda l'Appendice B per un completo elenco dei registri.

L'istruzione MOVE dell'esempio seguente imposta a $21000 il puntatore al primo bitplane e a $25000 il puntatore al secondo bitplane.[2]

```
DC.W  $00E0,$0002 ;Sposta $0002 nel registro $0E0 (BPL1PTH)
DC.W  $00E2,$1000 ;Sposta $1000 nel registro $0E2 (BPL1PTL)
DC.W  $00E4,$0002 ;Sposta $0002 nel registro $0E4 (BPL2PTH)
DC.W  $00E6,$5000 ;Sposta $5000 nel registro $0E6 (BPL2PTL)
```

Normalmente vengono inclusi gli appropriati file ".i", per poter usare i nomi simbolici al posto degli indirizzi numerici. Si raccomanda vivamente di seguire questa procedura, che rende più semplice adattare il software a eventuali future modifiche dell'hardware. Per esempio:

```
INCLUDE "hardware/custom.i"

DC.W  bplpt+$00,$0002 ;Sposta $0002 nel registro $0E0 (BPL1PTH)
DC.W  bplpt+$02,$1000 ;Sposta $1000 nel registro $0E2 (BPL1PTL)
DC.W  bplpt+$04,$0002 ;Sposta $0002 nel registro $0E4 (BPL2PTH)
DC.W  bplpt+$06,$5000 ;Sposta $5000 nel registro $0E6 (BPL2PTL)
```

Per utilizzare gli esempi di questo manuale, l'Appendice J contiene uno speciale file include che definisce i nomi dei registri basandosi sul file "hardware/custom.i". Questo per rendere gli esempi più leggibili da un punto di vista hardware. Gran parte degli esempi del libro hanno lo scopo di chiarire le spiegazioni, ma quasi nessuno è realmente utilizzabile senza modifiche e senza una grande quantità di codice aggiuntivo.

---

(1) Il prefisso $ serve per rappresentare i numeri esadecimali.

(2) Tutti i codici di esempio sono scritti in linguaggio Assembly.

L'istruzione WAIT

L'istruzione WAIT fa sì che il Copper entri in stato d'attesa fino a quando il pennello elettronico sullo schermo non raggiunge (o supera) le coordinate specificate nell'istruzione stessa. Durante l'attesa il Copper non utilizza né il bus né cicli di accesso alla memoria.

La prima word dell'istruzione contiene le coordinate orizzontali e verticali. La seconda contiene una maschera di bit di abilitazione per indicare al sistema quali bit della posizione del pennello elettronico vanno considerati nel confronto.

PRIMA WORD DELL'ISTRUZIONE (IR1)

Bit 0 Sempre impostato a 1.

Bit 15-8 Posizione verticale (VP).

Bit 7-1 Posizione orizzontale (HP).

SECONDA WORD DELL'ISTRUZIONE (IR2)

Bit 0 Sempre impostato a 0.

Bit 15 È il bit di attesa del Blitter. Generalmente è impostato a 1 (si veda il paragrafo "Informazioni avanzate").

Bit 14-8 Bit di abilitazione del confronto verticale (VE).

Bit 7-1 Bit di abilitazione del confronto orizzontale (HE).

La seguente istruzione WAIT attende la linea 150 ($96) ignorando la posizione orizzontale.

```
DC.W $9601,$FF00 ;Attende la linea 150, qualunque
                 ;sia la posizione orizzontale.
```

L'istruzione seguente, invece, attende la linea 255 e la posizione orizzontale 254. Questo è un evento che non può mai verificarsi, perciò il Copper rimane in attesa fino all'inizio del successivo intervallo di vertical blanking.

```
DC.W $FFFF,$FFFE ;Attende la linea 255, con posizione
                 ;orizzontale 254 (cioe' il Copper conclude
                 ;la sua lista di istruzioni).
```

Per capire perché la posizione VP=$FF e HP=$FE non può essere raggiunta, occorre esaminare le restrizioni che regolano il confronto. La linea 255 è una linea valida, anche se è il massimo valore raggiungibile dal contatore. Dal momento che 255 è il valore massimo, la linea successiva avrà valore zero (la linea 256 vale zero nel confronto). La posizione orizzontale, però, ha un valore massimo di 226 ($E2). Ciò significa che il massimo valore riscontrabile per la posizione del pennello elettronico è $FFE2.

Si potrebbe essere tentati d'indicare la posizione orizzontale $FE, che non può essere

raggiunta, ma un valore minore di $FF come posizione verticale. In questo modo, però, non si ottiene il risultato previsto. Infatti, il confronto ha un risultato positivo quando la posizione raggiunta è *maggiore* o uguale a quella richiesta. Se il valore di VP non fosse $FF, appena raggiunta la linea successiva a quella specificata il Copper uscirebbe dallo stato d'attesa.

Il prossimo paragrafo, che riguarda la posizione orizzontale e verticale del pennello elettronico, si applica sia all'istruzione WAIT sia all'istruzione SKIP. Quest'ultima viene trattata nel paragrafo "Informazioni avanzate".

POSIZIONE ORIZZONTALE DEL PENNELLO ELETTRONICO

Il valore orizzontale della posizione del pennello elettronico può variare da $0 a $E2. Il bit meno significativo non viene utilizzato nel confronto, e quindi le posizioni disponibili sono in totale 113. Ciò corrisponde a 4 pixel in bassa risoluzione e a 8 pixel in alta risoluzione. L'horizontal blanking avviene nell'intervallo $0F-$35. Lo schermo standard (di 320 pixel) ha un intervallo inutilizzato che va da $04 a $47, durante il quale è visibile solo il colore di fondo.

Nel sistema NTSC le linee non hanno tutte la medesima lunghezza. Una linea ogni due è una linea lunga (228 clock di colore, $0-$E3), mentre l'altra è lunga soltanto 227 clock di colore. Nel sistema PAL tutte le linee sono lunghe 227 clock di colore. Il display considera tutte le linee lunghe 227,5 clock di colore, mentre il Copper vede linee alternate corte e lunghe.

POSIZIONE VERTICALE DEL PENNELLO ELETTRONICO

La posizione verticale ha la risoluzione di una singola linea, con un valore massimo di 255. In realtà vi sono 312 posizioni in PAL (262 in NTSC), ma possono esservi alcune complicazioni se si desidera che accada qualcosa in una delle ultime linee. Dal momento che la posizione verticale è espressa su 8 bit (e permette quindi solo 256 valori) una delle possibili soluzioni è la seguente:

ISTRUZIONE	SPIEGAZIONE
[... altre istruzioni ...]	
Attendere (WAIT) la posizione (0,255).	*A questo punto sembra che il contatore verticale ritorni a zero.*
Attendere (WAIT) una qualunque posizione orizzontale da 0 a 55, che si riferisce quindi alle linee successive alla 255.	*In questo modo si raggiunge ogni linea compresa tra 256 e 256 + 56 = 312.*

Nota: anche verticalmente vi è un alternarsi di campi lunghi e corti (solo in modo interlace). In NTSC i campi sono di 262 e 263 linee, in PAL di 312 e 313.

Uno spostamento orizzontale occupa un ciclo di clock.

NTSC – 3.579.545 Hz
PAL – 3.546.895 Hz
Genlock – la frequenza di clock di base più o meno il 2%.

I BIT DI ABILITAZIONE DEL CONFRONTO

I bit 14-1 sono in genere impostati a 1. L'uso di questi bit viene descritto nel paragrafo "Informazioni avanzate".

Uso dei registri del Copper

Vi sono parecchi registri dedicati al Copper:

- Registri di locazione.
- Registri strobe di salto.
- Registro di controllo.

REGISTRI DI LOCAZIONE

Il Copper possiede due serie di registri di locazione:

COP1LCH	I 4 bit più significativi dell'indirizzo della prima lista di istruzioni.
COP1LCL	I 15 bit meno significativi dell'indirizzo della prima lista di istruzioni.
COP2LCH	I 4 bit più significativi dell'indirizzo della seconda lista di istruzioni.
COP2LCL	I 15 bit meno significativi dell'indirizzo della seconda lista di istruzioni.

Nell'accedere direttamente all'hardware, accade spesso di dover accedere a un indirizzo contenuto in una coppia di registri. Il registro con l'indirizzo più basso ha sempre al termine del nome una "H" e contiene i dati più significativi, quello con l'indirizzo più alto ha sempre una "L" al termine del nome e contiene i dati meno significativi. Per scrivere un indirizzo in tali registri è sufficiente scrivere una long word all'indirizzo del registro terminante con "H". Infatti, quando il 68000 scrive una long word ne colloca la word più significativa nell'indirizzo più basso.

Nel caso dei registri del Copper, vengono usati COP1LCH e COP2LCH. In seguito, per semplicità, questi indirizzi saranno chiamati COP1LC e COP2LC.

I registri di locazione del Copper contengono due indirizzi di salto indiretto. Il Copper legge le sue istruzioni tramite un program counter interno, che viene incrementato a ogni lettura. Quando si verifica un accesso in scrittura a uno dei registri strobe, l'indirizzo contenuto nel corrispondente registro di locazione viene caricato nel program counter. Questo fa sì che il Copper salti a una nuova locazione, dalla quale sarà estratta l'istruzione successiva. L'esecuzione delle istruzioni continua quindi sequenzialmente fino a quando il Copper non viene interrotto da un altro accesso a un indirizzo strobe.

All'inizio di ogni intervallo di vertical blanking, il sistema usa automaticamente COP1LC per reinizializzare il program counter del Copper. In pratica, qualunque cosa stesse facendo il Copper, esso viene forzato a ricominciare le sue operazioni dall'indirizzo contenuto in COP1LC.

REGISTRI STROBE DI SALTO

Accedendo in scrittura a uno di questi registri, il Copper viene forzato a ricaricare il proprio program counter in base al registro di locazione corrispondente. Il Copper stesso può scrivere in questi registri per eseguire salti programmati. Per esempio, si può usare l'istruzione MOVE per impostare a un determinato indirizzo il registro COP2LC. In seguito, qualunque istruzione MOVE relativa all'indirizzo COPJMP2 caricherà questo indirizzo nel program counter.

Vi sono due registri strobe di salto:

COPJMP1	Fa ripartire il Copper dall'indirizzo contenuto in COP1LC.
COPJMP2	Fa ripartire il Copper dall'indirizzo contenuto in COP2LC.

IL REGISTRO DI CONTROLLO

Ad alcuni registri il Copper può accedere sempre, ad altri solo quando uno speciale bit di controllo è impostato a 1, ad altri ancora non può accedere mai. La prima categoria comprende i registri da $80 in poi. Quelli inaccessibili sono invece quelli che vanno da $0 a $3E incluso. Il registro di controllo del Copper fa parte di quest'ultimo gruppo. Perciò è necessaria un'azione deliberata da parte del 68000 perché il Copper possa accedere al terzo gruppo di registri, quelli compresi tra $40 e $7E.

Il registro di controllo, chiamato COPCON, contiene un solo bit, il bit #1. Questo bit, chiamato CDANG, protegge tutti i registri che vanno da $40 a $7E. Questo intervallo comprende i registri di controllo del Blitter. Quando CDANG è a zero, questi registri non sono modificabili dal Copper. Ciò serve a prevenire situazioni in cui, a causa di errori nella lista delle istruzioni, il Copper tramite il Blitter modifica in maniera incontrollata la memoria del sistema.

Il bit CDANG viene azzerato a ogni reset.

Costruzione di una lista di istruzioni del Copper

La lista del Copper contiene tutte le istruzioni necessarie all'inizializzazione dei registri durante l'intervallo di vertical blanking e quelle per modificare i registri durante la visualizzazione dell'immagine video. Può essere più semplice avvicinarsi all'argomento pensando a ogni elemento come a un sotto-sistema separato: playfield, sprite, audio, interrupt e così via. Si creano quindi tante liste di istruzioni separate per ogni sotto-sistema.

Quando queste liste parziali sono complete, vengono fuse in un'unica lista che il Copper esegue a ogni quadro video. L'alternativa è quella di creare direttamente la lista finale.

Per esempio, i puntatori ai bitplane che costituiscono i playfield e i puntatori agli sprite devono essere reinizializzati a ogni intervallo di vertical blanking perché il nuovo display venga tracciato con i dati corretti. A questo scopo è necessaria una lista di istruzioni come la seguente:

WAIT la prima linea del display
MOVE i dati relativi al puntatore al primo bitplane
MOVE i dati relativi al puntatore al secondo bitplane
MOVE i dati relativi al primo sprite
e così via...

Ecco un altro esempio. I canali DMA degli sprite possono essere utilizzati più volte nello stesso quadro video, cambiando la forma e le dimensioni dello sprite (mentre in genere i colori vengono mantenuti inalterati). Anche i colori però possono essere cambiati tramite una lista di istruzioni del Copper che attenda l'ultima linea del primo uso di uno sprite e che modifichi i colori prima dell'uso successivo:

```
WAIT la prima linea del display
MOVE il colore1.1 in COLOR17
MOVE il colore1.2 in COLOR18
MOVE il colore1.3 in COLOR19
WAIT l'ultima linea + 1 del primo uso dello sprite
MOVE il colore2.1 in COLOR17
MOVE il colore2.2 in COLOR18
MOVE il colore2.3 in COLOR19
e così via...
```

Nella creazione delle liste di istruzioni del Copper, si deve tener presente che la lista finale dev'essere ordinata secondo l'ordine in cui il pennello elettronico percorre lo schermo. Esso parte dalla posizione (0,0) nell'angolo superiore sinistro fino a raggiungere l'angolo inferiore destro: (226,312) in PAL o (226,262) in NTSC. Ciò significa che un'istruzione relativa alla posizione (0,100) deve venire *dopo* un'istruzione relativa alla posizione (0,60).

A causa della particolare struttura dell'istruzione WAIT, ci sono alcune eccezioni a questa regola. L'istruzione WAIT, infatti, attende che la posizione sia uguale *o maggiore* di quella indicata nell'istruzione.

Ciò significa, per esempio, che se vi sono due istruzioni successive del genere:

```
WAIT (64,64)
MOVE dati

WAIT (60,60)
MOVE dati
```

il Copper eseguirà *entrambe* le istruzioni MOVE anche se queste non sono in sequenza corretta. La specifica "maggiore o uguale" impedisce al Copper di bloccarsi se la posizione richiesta è già stata superata. Anche la seconda istruzione MOVE dell'esempio seguente viene eseguita:

```
WAIT (60,60)
MOVE dati

WAIT (60,60)
MOVE dati
```

Al momento della seconda istruzione WAIT, il contatore avrà già superato la posizione specificata e perciò verrà mandata in esecuzione la seconda istruzione MOVE.

La precedente sequenza di istruzioni è equivalente alla più semplice:

```
WAIT (60,60)
MOVE dati
MOVE dati
```

dal momento che ogni istruzione WAIT può essere seguita anche da più di una istruzione MOVE.

UN ESEMPIO COMPLETO DI LISTA DI ISTRUZIONI

Il seguente esempio mostra una completa lista di istruzioni del Copper. È una lista per due bitplane, uno all'indirizzo $21000 e l'altro all'indirizzo $25000. All'inizio dello schermo i registri di colore vengono impostati con i seguenti valori:

Registro	Colore
COLOR00	bianco
COLOR01	rosso
COLOR02	verde
COLOR03	blu

Alla linea 150, nei registri vengono inseriti nuovi colori:

Registro	Colore
COLOR00	nero
COLOR01	giallo
COLOR02	cyan
COLOR03	magenta

La seguente è la lista completa.

```
;Note:
;       1. Le istruzioni devono risiedere nella memoria chip.
;       2. Gli indirizzi dei bitplane usati sono solo un esempio.
        3. Gli indirizzi dei registri sono espressi nelle
           istruzioni del Copper come offset dall'indirizzo
           base dei chip custom.
;       4. L'esempio presuppone che il programma abbia assunto
;          in precedenza il pieno controllo dell'hardware e che
           non sia in conflitto con il sistema operativo.
        5. Molti esempi accedono direttamente agli indirizzi
;          di memoria loro necessari. Normalmente occorre allocare
;          la memoria tramite l'istruzione AllocMem o equivalenti.
;       6. Questi esempi hanno uno scopo didattico, non pratico.
;       7. I seguenti file include sono necessari a tutti gli
;          esempi di questo capitolo.


        INCLUDE "exec/types.i"
        INCLUDE "hardware/custom.i"
        INCLUDE "hardware/dmabits.i"
        INCLUDE "hardware/hw_examples.i"

COPPERLIST:
;
;Prepara i puntatori ai due bitplane
```

```
      DC.W BPL1PTH,$0002 ;Sposta $0002 nel registro $0E0 (BPL1PTH)
      DC.W BPL1PTL,$1000 ;Sposta $1000 nel registro $0E2 (BPL1PTL)
      DC.W BPL2PTH,$0002 ;Sposta $0002 nel registro $0E4 (BPL2PTH)
      DC.W BPL2PTL,$5000 ;Sposta $5000 nel registro $0E6 (BPL2PTL)
;
;Prepara i registri di colore
;
      DC.W COLOR00,$0FFF ;Sposta il bianco nel registro $180 (COLOR00)
      DC.W COLOR01,$0F00 ;Sposta il rosso nel registro $182 (COLOR01)
      DC.W COLOR02,$00F0 ;Sposta il verde nel registro $184 (COLOR02)
      DC.W COLOR03,$000F ;Sposta il blu nel registro $186 (COLOR03)
;
;Specifica due bitplane in bassa risoluzione
;
      DC.W BPLCON0,$2200 ;due bitplane in bassa risoluzione
;
;Attende la linea 150
;
      DC.W $9601,$FF00   ;Attende la linea 150, ignora la posizione orizzontale
;
;Cambia i colori a meta' schermo
;
      DC.W COLOR00,$0000 ;Sposta il nero nel registro $180 (COLOR00)
      DC.W COLOR01,$0FF0 ;Sposta il giallo nel registro $182 (COLOR01)
      DC.W COLOR02,$00FF ;Sposta il cyan nel registro $184 (COLOR02).
      DC.W COLOR03,$0F0F ;Sposta il magenta nel registro $186 (COLOR03)
;
;Termina la lista attendendo un evento impossibile
;
        DC.W $FFFF,$FFFE   ;Attende la linea 255, H = 254
```

Per ulteriori informazioni riguardanti i registri di colore, si veda il Capitolo 3.

LOOP E SALTI CONDIZIONATI

Loop e salti condizionati all'interno delle liste del Copper sono trattati nel paragrafo "Informazioni avanzate".

Avvio e arresto del Copper

AVVIO DEL COPPER DOPO UN RESET

All'accensione o al reset del sistema, uno dei registri di locazione del Copper (COP1LC o COP2LC) dev'essere inizializzato e occorre accedere al corrispondente indirizzo strobe prima che venga abilitato il canale DMA del Copper. Ciò assicura uno stato iniziale conosciuto. In genere si utilizza COP1LC, dal momento che viene automaticamente reinizializzato a ogni intervallo di vertical blanking. La seguente sequenza di istruzioni mostra come inizializzare un registro di

locazione. Si presume che il programma abbia già creato una lista di istruzioni all'indirizzo "coplist".

```
;
;Installa la lista di istruzioni
;
      LEA    CUSTOM,a1       ;a1 = indirizzo dei chip custom
      LEA    coplist(pc),a0  ;a0 = indirizzo della lista
      MOVE.L a0,COP1LC(a1)   ;Scrive l'indirizzo
      MOVE.W d0,COPJMP(a1)   ;Provoca il reset del program counter
;
;Questa istruzione abilita il DMA del Copper e dei bitplane

      MOVE.W #(DMAF_SETCLR!DMAF_COPPER!DMAF_RASTER!DMAF_MASTER),DMACON(a1)
```

Ora, se il contenuto di COP1LC non cambia, a ogni intervallo di vertical blanking il Copper viene forzato a rieseguire la medesima sequenza di istruzioni. Si forma così un ciclo continuo che, con una lista formulata correttamente, produce un'immagine video stabile.

ARRESTO DEL COPPER

Non esistono istruzioni specifiche per l'arresto del Copper. Per essere sicuri che non esegua ulteriori istruzioni fino al termine del display, quando il program counter riparte dall'inizio della lista, l'ultima istruzione dev'essere un'istruzione WAIT con una condizione che non si può verificare. L'istruzione tipica in questi casi è WAIT VP = $FF e HP = $FE. Un valore di HP maggiore di $E2 non può essere raggiunto. Quando inizia il successivo intervallo di vertical blanking, il Copper viene forzato a ripartire dall'inizio della lista, e l'istruzione WAIT non viene completata.

Il Copper può essere fermato anche disabilitando il suo canale DMA e impedendogli quindi di accedere alle sue istruzioni. Il registro DMACON contiene tutti i bit relativi al controllo del DMA. Qiando il bit 7 (COPEN) è impostato a 1, il Copper può accedere al canale DMA. Per ulteriori informazioni riguardo al DMA si veda il Capitolo 7.

Informazioni avanzate

L'ISTRUZIONE SKIP

L'istruzione SKIP fa sì che il Copper non esegua l'istruzione successiva, se il contatore di posizione del pennello elettronico è uguale o superiore al valore specificato nell'istruzione stessa.

Di seguito è illustrato il formato delle word che costituiscono l'istruzione SKIP. È uguale a quello dell'istruzione WAIT, con l'eccezione del bit 0 della seconda word che dev'essere a 1 per identificarla come istruzione SKIP.

PRIMA WORD DELL'ISTRUZIONE (IR1)

Bit 0 Sempre impostato a 1.

Bit 15-8 Posizione verticale (VP).

Bit 7-1 Posizione orizzontale (HP).

SECONDA WORD DELL'ISTRUZIONE (IR2)

Bit 0 Sempre impostato a 1.

Bit 15 Il bit di attesa del Blitter.

Bit 14-8 Bit di abilitazione del confronto verticale (VE).

Bit 7-1 Bit di abilitazione del confronto orizzontale (HE).

Per quanto riguarda la posizione orizzontale e verticale, si applicano anche qui le osservazioni fatte per l'istruzione WAIT.

L'istruzione SKIP dell'esempio seguente salta l'istruzione successiva se il valore di VP è maggiore o uguale a 100 ($64).

```
DC.W $6401,$FF01 ;Se VP ≥ 100 salta l'istruzione successiva
                 ;ignorando HP.
```

LOOP, SALTI CONDIZIONATI E ABILITAZIONE DEL CONFRONTO

In ogni momento i valori degli indirizzi contenuti nei registri di locazione possono essere cambiati per eseguire dei loop nella lista delle istruzioni. Prima che si verifichi il successivo intervallo di vertical blanking, tuttavia, il registro COP1LC *deve* essere riportato al valore corrispondente all'inizio della lista di istruzioni. Questo valore sarà impostato nel program counter del Copper all'inizio del successivo intervallo di vertical blanking.

I bit da 14 a 1 della seconda word delle istruzioni WAIT e SKIP specificano quali bit devono essere usati nel confronto, tra quelli che descrivono la posizione. Prima d'intraprendere qualsiasi altra azione, la posizione nella word 1 e i bit corrispondenti della word 2 vengono confrontati con il valore contenuto nel registro di controllo della posizione del pennello elettronico. Un determinato bit nella word 1 entra nel confronto *se e solo se* il corrispondente bit di abilitazione della word 2 è impostato a 1. Se invece il bit è a 0, il confronto per quel particolare bit viene considerato sempre vero. Per esempio, se si è interessati solamente al valore degli ultimi quattro bit della posizione verticale, dovranno essere impostati a 1 soltanto gli ultimi 4 bit di abilitazione (11-8) nella seconda word dell'istruzione.

Non tutti i bit del registro di locazione possono essere mascherati in questo modo. Osservando la descrizione della seconda word delle istruzioni WAIT e SKIP si può notare che il bit 15 è il bit di attesa del Blitter (BFD, per Blitter-finished-disable). Questo bit non fa parte della maschera di confronto ma ha un suo particolare significato per l'istruzione WAIT. Perciò, il bit più significativo delle istruzioni WAIT e SKIP non può essere mascherato. Questa limitazione in genere non pone alcun problema pratico, l'esempio seguente, tuttavia, ne mostra un possibile caso.

Questo esempio fa sì che il Copper generi un interrupt ogni 16 linee di scansione. Può sembrare che il modo corretto di ottenere questo risultato sia quello di usare una maschera di valore $0F e di confrontare quindi il risultato con il valore $0F. Ciò dovrebbe portare a un risultato "vero" per $1F, $2F, $3F... Dal momento che il test ha una condizione di maggiore o uguale, *sembra* che venga così abilitato il confronto ogni 16 linee. Tuttavia, come si è visto, il bit più significativo della posizione non può essere mascherato e apparirà comunque nei confronti. Se il Copper sta attendendo la posizione $0F e la posizione verticale ha superato il valore 128 ($80), il test sarà sempre verificato e l'interrupt si verificherà a ogni linea di scansione (questo dipende sempre dal fatto che il Copper è sottoposto a una condizione di maggiore o uguale).

Nell'esempio seguente, la lista di istruzioni del Copper è strutturata in modo da produrre un loop. I valori di COP1LC e COP2LC devono essere inizializzati dal 68000 o dal Copper stesso *prima* di questa sezione di codice. Inoltre, si presume che il server relativo all'interrupt generato ogni 16 linee sia già stato correttamente installato. Si noti inoltre che si tratta di linee di scansione in modo non interlace.

I due loop sono in gran parte uguali. In entrambi il Copper attende che la posizione raggiunga il valore $?F (dove ? è un qualunque numero esadecimale), e genera un interrupt. Per essere sicuri che il Copper non riesegua il loop due volte nella stessa linea e che quindi generi un altro interrupt si entra in attesa della posizione orizzontale $E2 dopo ogni interrupt. La posizione $E2 corrisponde all'ultima posizione raggiungibile di una riga. Operando in questo modo, la successiva istruzione WAIT non viene eseguita prima della linea successiva. Il loop viene eseguito accedendo al registro COPJMP1. Ciò fa sì che il Copper salti all'indirizzo contenuto in COP1LC.

Il problema di mascheratura visto in precedenza, però, impedisce a questo sistema di funzionare correttamente dopo la linea 127. Dopo questa linea deve quindi essere eseguito un secondo loop. Quando la posizione verticale diventa maggiore o uguale a 127, l'istruzione che causa il primo loop viene saltata tramite un'istruzione SKIP, facendo procedere il Copper verso il secondo loop. Questo loop è molto simile al primo, con la differenza che il valore d'attesa dell'istruzione WAIT ha il bit più alto impostato (in binario 1xxx1111). Questo vale sia per la condizione di attesa orizzontale sia per quella verticale. Per provocare questo loop si accede al registro COPJMP2. La lista viene messa in attesa quando VP ≥ 255, in modo che termini prima dell'intervallo di vertical blanking. A questo punto COP1LC viene riscritto dal sistema operativo, e l'esecuzione ricomincia dal primo loop.

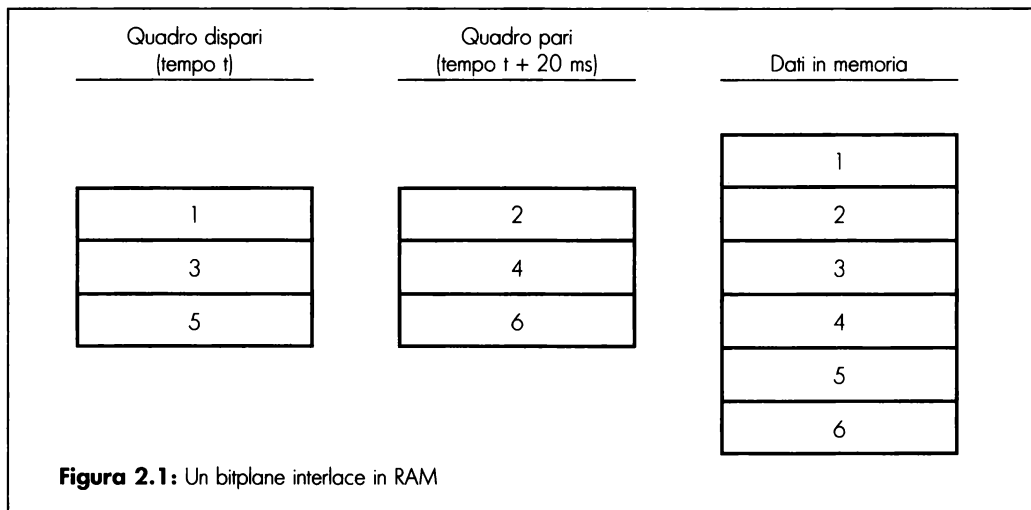
Il registro COP1LC viene inizializzato al termine dell'intervallo di vertical blanking da un handler di interrupt che fa parte della catena di server relativa all'interrupt di vertical blanking. Se questa catena di server non viene alterata, COP1LC viene correttamente ricaricato nel program counter del Copper al termine dell'intervallo di vertical blanking.

```
;
;Questi sono i dati della lista di istruzioni.

;Si presume che COPPERL1 e COPPERL2 siano stati caricati in COP1LC
;e COP2LC in precedenza.
;
COPPERL1:
      DC.W $0F01,$8F00   ;WAIT VP = 0xxx1111
      DC.W INTREQ,$8010  ;Richiede l'interrupt
      DC.W $00E3,$80FE   ;WAIT HP = $E2
      DC.W $7F01,$7F01   ;SKIP se VP ≥ 127
      DC.W COPJMP1,$0    ;Salta a COP1LC
```

```
COPPERL2:
      DC.W $8F01,$8F00   ;WAIT VP = 1xxx1111
      DC.W INTREQ,$8010 ;Richiede l'interrupt
      DC.W $80E3,$80FE   ;WAIT HP = $E2
      DC.W $FF01,$FE01   ;SKIP se VP ≥ 255
      DC.W COPJMP2,$0    ;Salta a COP2LC

      DC.W $FFFF,$FFFE   ;Fine della lista di istruzioni
```

USO DEL COPPER IN MODO INTERLACE

Un bitplane interlace ha una risoluzione verticale doppia rispetto al normale. Mentre un display PAL normale possiede 312 linee, uno interlace ne possiede 625. In modo interlace, il pennello elettronico scandisce lo schermo due volte dall'alto in basso mostrando alternativamente 312 e 313 linee. Durante la prima scansione vengono visualizzate le linee dispari, nella seconda quelle pari. La circuiteria tratta quindi un display interlace come due quadri separati, uno contenente linee dispari, l'altro linee pari. La Figura 2.1 mostra come i relativi dati si

Figura 2.1: Un bitplane interlace in RAM

presentano in memoria.

Il sistema preleva i dati per la visualizzazione tramite puntatori all'indirizzo iniziale dei dati. Come si vede, l'indirizzo dei dati relativi alle linee pari supera di una linea quello relativo alle linee dispari. Il puntatore al bitplane dovrà perciò avere un valore diverso nei due quadri video.

L'organizzazione dei dati in memoria ricalca da vicino quella dell'immagine sullo schermo, cioè le linee pari e dispari sono intercalate fra loro. Ciò si ottiene tramite due liste separate di istruzioni Copper per ognuno dei due quadri video.

Per far sì che il Copper alterni le due liste, occorre predisporre un interrupt del 68000 subito dopo la prima linea di schermo. Quando l'interrupt viene eseguito si cambia il contenuto di COP1LC in modo che punti alla seconda lista. Durante il successivo intervallo di vertical blanking, il Copper torna automaticamente alla prima lista.

Per ulteriori informazioni riguardanti i display in interlace, si veda il Capitolo 3.

USO DEL COPPER CON IL BLITTER

Se il Copper viene utilizzato per dare il via a una serie di operazioni del Blitter, ha l'obbligo di aspettare che il Blitter abbia concluso il suo compito prima d'iniziare una nuova operazione. Modificare i registri del Blitter durante il suo funzionamento può portare a risultati imprevedibili. A questo scopo l'istruzione WAIT include un bit supplementare, chiamato BFD (Blitter Finished Disable). In genere, questo bit è a 1 e l'istruzione WAIT è condizionata solo dalla posizione del pennello elettronico.

Quando BFD viene azzerato, la logica dell'istruzione WAIT viene modificata. Il Copper attende fino a quando il valore della posizione non è maggiore o uguale a quanto richiesto e attende anche che il Blitter abbia terminato il suo compito. Questo bit dev'essere usato con cautela. In alcuni casi azzerarlo impedisce la visualizzazione di alcuni schermi o di altri oggetti video.

Per ulteriori informazioni riguardo al Blitter, si veda il Capitolo 6.

IL COPPER E IL 68000

Quando le capacità del Copper non sono sufficienti, è possibile interrompere il lavoro del 68000 per utilizzare il suo set di istruzioni. Per farlo è sufficiente usare l'istruzione MOVE del Copper per impostare a 1 i seguenti bit del registro INTREQ:

Tavola 2.1: Interruzione del 68000

Numero del bit	**Nome**	**Funzione**
15	SET/CLR	Bit di controllo. Determina se i bit a 1 devono essere impostati o azzerati.
4	COPEN	Interrupt generato dal Copper.

Per ulteriori informazioni riguardo agli interrupt si consulti il Capitolo 7.

Sommario delle istruzioni del Copper

La tavola seguente mostra il sommario delle posizioni dei singoli bit per ognuna delle istruzioni del Copper. Per un elenco completo dei registri si veda l'Appendice A.

Tavola 2.2: Sommario delle istruzioni del Copper

	MOVE		WAIT		SKIP	
Bit	**IR1**	**IR2**	**IR1**	**IR2**	**IR1**	**IR2**
15	X	RD15	VP7	BFD	VP7	BFD
14	X	RD14	VP6	VE6	VP6	VE6
13	X	RD13	VP5	VE5	VP5	VE5
12	X	RD12	VP4	VE4	VP4	VE4
11	X	RD11	VP3	VE3	VP3	VE3
10	X	RD10	VP2	VE2	VP2	VE2
09	X	RD09	VP1	VE1	VP1	VE1
08	DA8	RD08	VP0	VE0	VP0	VE0
07	DA7	RD07	HP8	HE8	HP8	HE8
06	DA6	RD06	HP7	HE7	HP7	HE7
05	DA5	RD05	HP6	HE6	HP6	HE6
04	DA4	RD04	HP5	HE5	HP5	HE5
03	DA3	RD03	HP4	HE4	HP4	HE4
02	DA2	RD02	HP3	HE3	HP3	HE3
01	DA1	RD01	HP2	HE2	HP2	HE2
00	0	RD00	1	0	1	1

X = Ininfluente, impostare a zero per compatibilità con sistemi futuri
IR1 = Prima word dell'istruzione
IR2 = Seconda word dell'istruzione
DA = Registro destinazione
RD = Dati da muovere nel registro destinazione
VP = Posizione verticale
HP = Posizione orizzontale
VE = Bit di abilitazione verticale
HE = Bit di abilitazione orizzontale
BFD = Bit di attesa del Blitter

Introduzione

L'immagine video generata dall'Amiga consiste di due parti principali: i playfield, chiamati talvolta sfondi, e gli sprite, ovvero oggetti grafici che si possono spostare sullo schermo. Questo capitolo descrive come si accede ai registri hardware per creare e controllare i playfield.

PRESENTAZIONE DEL CAPITOLO

Il capitolo inizia con una breve panoramica delle principali caratteristiche dei playfield e con la definizione di alcuni termini fondamentali. Prosegue poi con i seguenti argomenti:

- La formazione di un "normale" playfield. Un playfield cioè delle stesse dimensioni dello schermo. Questa sezione contiene i concetti generali relativi alla creazione dei playfield.

- La creazione di uno schermo dual-playfield, in cui un playfield si sovrappone a un altro. Questo procedimento differisce in alcuni dettagli dal precedente.

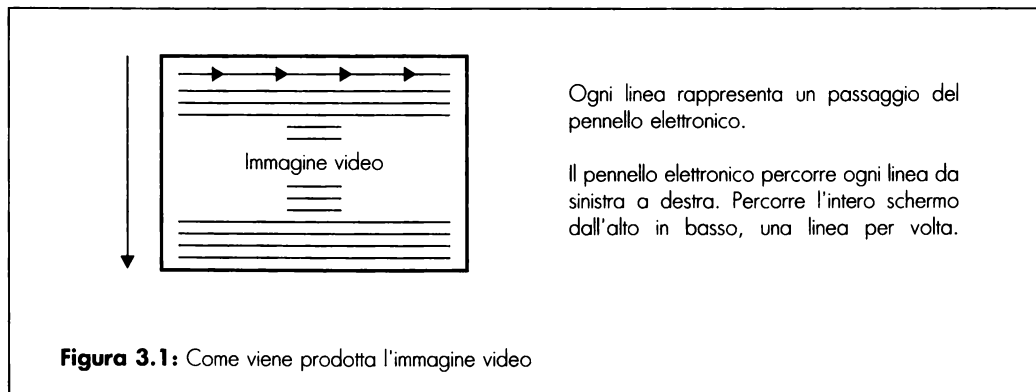
- La creazione di playfield aventi diverse dimensioni, e la visualizzazione parziale di un playfield più grande dello schermo.

- Il movimento orizzontale e verticale dei playfield.

- Informazioni avanzate riguardanti il controllo dei playfield in particolari situazioni.

CARATTERISTICHE DEI PLAYFIELD

L'Amiga produce il suo display con una tecnica di tipo raster: l'immagine visibile sullo schermo è costituita da una serie di linee orizzontali generate una dopo l'altra. Ogni linea orizzontale è formata da una serie di pixel. Un'immagine grafica viene creata definendo in memoria uno o più bitplane e riempiendoli di valori 1 e 0. La combinazione di questi valori determina i colori e quindi l'aspetto dell'immagine.

Figura 3.1: Come viene prodotta l'immagine video

Il pennello elettronico produce circa 312 linee in un sistema PAL, ma ne sono normalmente visibili solo 256. In NTSC questi valori sono rispettivamente 262 e 200. Ogni serie completa di 312 (o 262) linee viene chiamata "quadro video". Il tempo di quadro, ovverosia il tempo necessario alla generazione completa di un quadro, è di circa 1/50 di secondo in un sistema PAL e di 1/60 in NTSC. Tra un quadro e il successivo, il pennello elettronico attraversa le linee non visibili sullo schermo e ritorna al punto di partenza per generare il quadro seguente.

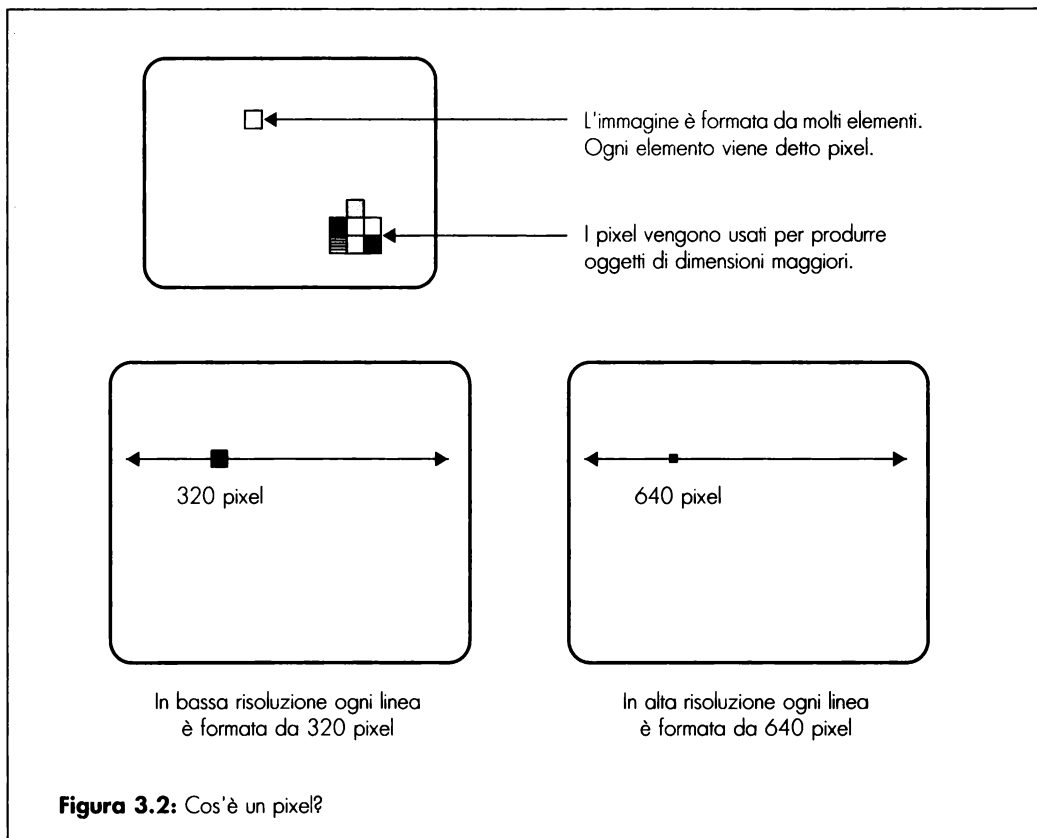
L'area dello schermo è definita come un reticolo di pixel. Un pixel è un singolo elemento grafico, la più piccola parte indirizzabile di un'immagine video. La Figura 3.2 spiega che cos'è un pixel e come fa un insieme di pixel a formare il quadro video.

L'Amiga offre la possibilità di scegliere sia la risoluzione orizzontale sia quella verticale. La risoluzione orizzontale può essere bassa o alta. La risoluzione verticale può essere interlace o non interlace.

- In bassa risoluzione un normale display ha una larghezza di 320 pixel.
- L'alta risoluzione consente invece di arrivare a 640 pixel orizzontali.
- In modo non interlace, la normale altezza di uno schermo è di 256 pixel (PAL) o 200 pixel (NTSC).
- In modo interlace, questa risoluzione viene raddoppiata a 512 (PAL) o 400 (NTSC) pixel.

Queste risoluzioni possono essere combinate fra loro in modo da avere, per esempio, un display in alta risoluzione interlace di 640 x 512 pixel.

Si noti che le dimensioni date in precedenza sono solo nominali, e rappresentano le dimensioni normalmente usate nei programmi. In realtà è possibile visualizzare anche playfield

Figura 3.2: Cos'è un pixel?

più grandi; se ne parla nel paragrafo "Bitplane e finestre video di diverse misure". Inoltre, le dimensioni di un playfield in memoria possono essere maggiori di quelle visualizzabili sullo schermo. È possibile scegliere quale parte deve apparire sullo schermo specificando una diversa dimensione per la finestra video.

Un playfield più alto dello schermo può essere spostato in alto e in basso. Un playfield più largo può essere spostato a destra e a sinistra. Questo tipo di scroll viene discusso nel paragrafo "Movimento dei playfield".

Nell'Amiga, in condizioni normali, si possono avere fino a 32 colori diversi sullo schermo. È possibile controllare il colore di ogni singolo pixel impostando o azzerando il bit o i bit che lo controllano. Un display creato in questo modo viene detto "bitmap".

Per esempio, in un display a due colori il colore di ogni pixel è determinato dal singolo bit a esso associato. Se questo bit è a 0 viene usato un colore, se è a 1 ne viene usato un altro. Un display a quattro colori ha invece bisogno di due bitplane. Al momento della visualizzazione, i due bitplane vengono sovrapposti, il che significa che a ogni pixel sono associati due bit. In questo modo si possono combinare fino a cinque bitplane, in condizioni normali. Playfield formati da tre, quattro o cinque bitplane permettono di ottenere rispettivamente otto, sedici e trentadue colori.

Il colore di un pixel è sempre determinato dalla combinazione binaria dei bit che lo definiscono. Quando il sistema combina fra loro i bitplane per dare origine all'immagine video, la combinazione dei bit per ogni pixel viene a corrispondere al numero di un registro di colore. L'Amiga ha 32 registri di colore, ognuno dei quali definisce un qualsiasi colore tratto da una

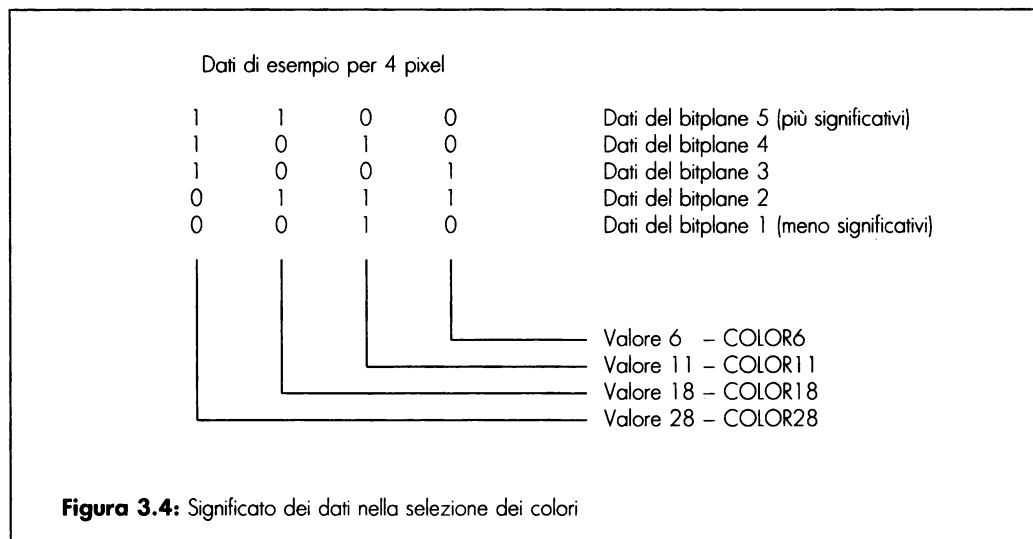
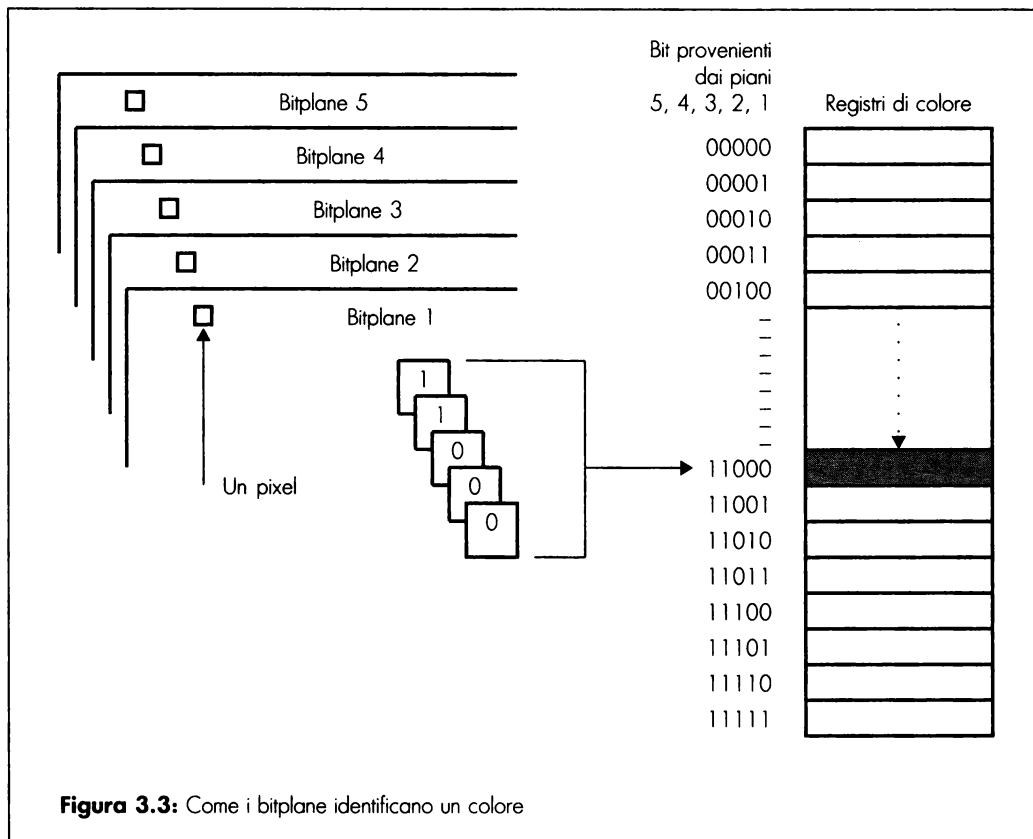
Bitplane 5
Bitplane 4
Bitplane 3
Bitplane 2
Bitplane 1
Un pixel
1
1
0
0
0

Bit provenienti dai piani 5, 4, 3, 2, 1

Registri di colore

00000
00001
00010
00011
00100
11000
11001
11010
11011
11100
11101
11110
11111

Figura 3.3: Come i bitplane identificano un colore

Figura 3.4: Significato dei dati nella selezione dei colori

palette di 4096.

La Figura 3.3 mostra in che modo i bitplane formano il codice che seleziona il registro di colore da usare per un particolare pixel.

I valori dei bitplane più alti sono quelli più significativi nella formazione di questi numeri. Come risulta dalla Figura 3.4, il valore di ogni pixel nel bitplane più alto corrisponde alla cifra più a sinistra del numero. Il valore nel bitplane inferiore dà origine alla cifra seguente e così via.

Vi è anche la possibilità di creare due playfield separati, ognuno formato da un massimo di tre bitplane. Ogni playfield utilizza una propria serie di colori. Questo modo è detto dual-playfield.

Creazione di un playfield

Per una graduale introduzione dell'argomento, questo paragrafo descrive come accedere ai registri hardware per formare un semplice playfield delle stesse dimensioni di una normale finestra video. Ciò dà origine a un bordo di un certo spessore tra il playfield e le estremità fisiche dello schermo. I playfield, in genere, non sfruttano tutta l'area fino all'orlo fisico dello schermo.

Per creare un playfield occorre definirne le seguenti caratteristiche:

- Larghezza e altezza del playfield e dimensioni della finestra video, cioè della parte che apparirà effettivamente sullo schermo.
- Il colore di ogni pixel del playfield.
- La risoluzione orizzontale.
- La risoluzione verticale, o interlace.
- La posizione dei dati in memoria e le loro dimensioni, perché il sistema sappia da dove prendere i dati da visualizzare e quanti visualizzarne per ogni linea.

Inoltre sarà necessario allocare la memoria per i dati del playfield e (opzionalmente) scrivere una lista di istruzioni del Copper che ne controlli la visualizzazione.

DIMENSIONI DEL PLAYFIELD

Per creare un playfield che abbia le medesime dimensioni dello schermo si usa un valore di 320 o 640 pixel per linea, a seconda della risoluzione orizzontale adottata. L'altezza può essere di 256 o 512 pixel in PAL oppure 200 o 400 in NTSC a seconda che venga scelto o meno il modo interlace.

BITPLANE E COLORE

Per definire i colori di un playfield si devono compiere le seguenti operazioni:

- Decidere quanti colori sono necessari e quale dev'essere il colore di ogni pixel.
- Caricare nei registri di colore i colori scelti.

- Allocare la memoria necessaria per i bitplane scelti, e predisporre per ognuno un puntatore.
- Scrivere le istruzioni necessarie a far sì che ogni pixel del playfield assuma il colore desiderato.

La Tavola 3.1 mostra quanti bitplane sono necessari per una certa selezione di colori.

Tavola 3.1: Colori di un playfield

Numero di colori	Numero di bitplane
2	1
3-4	2
5-8	3
9-16	4
17-32	5

La tavola dei colori

La tavola dei colori comprende 32 registri, e ciascuno può contenere un colore diverso. Ecco un estratto della tavola:

Tavola 3.2: Estratto della tavola dei colori

Registro	Contenuto	Significato
COLOR00	12 bit	Colore associato all'area di sfondo e ai bordi
COLOR01	12 bit	Colore numero 1 (per esempio, il secondo colore di un playfield a due colori)
COLOR02	12bit	Colore numero 2
COLOR31	12 bit	Colore numero 31

COLOR00 è sempre riservato al colore di fondo. Il colore di fondo è visibile nei bordi e in ogni area dello schermo in cui non sia presente alcun oggetto grafico.

Se si utilizza un'interfaccia genlock per l'input video da una telecamera, da un videoregistratore o da un disco laser, il colore di fondo sarà sostituito dall'immagine proveniente dall'apparecchiatura esterna.

I dodici bit dedicati alla selezione del colore permettono di scegliere, per ognuno dei 32 registri, uno dei possibili 4096 colori (si veda la Tavola 3.3).

Tavola 3.3: Contenuto dei registri di colore

Bit	Significato
Bit 15-12	Non utilizzati
Bit 11-8	Rosso
Bit 7-4	Verde
Bit 3-0	Blu

La Tavola 3.4 elenca alcuni valori d'esempio per i registri di colore, e il colore corrispondente. Al termine del capitolo si trova una lista più completa.

Tavola 3.4: Esempio di valori dei registri di colore

Contenuto del registro	Colore risultante
$FFF	Bianco
$6FE	Blu cielo
$DB9	Beige chiaro
$000	Nero

Proponiamo ora alcune istruzioni d'esempio su come caricare i valori corretti nei registri di colore:

```
LEA CUSTOM,a0          ;L'indirizzo base dei chip custom
MOVE.W #$FFF,COLOR00(a0) ;Carica il bianco nel registro 0
MOVE.W #$6FE,COLOR01(a0) ;Carica il blu cielo nel registro 1
```

I registri di colore sono a sola scrittura. È possibile conoscerne il contenuto solo osservando lo schermo. È buona norma, quindi, per questi come per gli altri registri a sola scrittura, tenere in RAM delle copie dei dati, e aggiornarle ogni volta che questi ultimi vengono modificati. In tal modo è sempre possibile conoscere, seppure indirettamente, il contenuto dei registri.

Selezione del numero di bitplane

Dopo avere deciso quanti colori sono necessari e quanti bitplane occorrono per ottenerli, è necessario informarne il sistema.

Questa selezione avviene impostando gli appropriati bit del registro BPLCON0 (BitPlane Control Register 0). I bit relativi sono i numeri 14, 13 e 12, denominati BPU2, BPU1 e BPU0. La Tavola 3.5 ne chiarisce il significato.

Tavola 3.5: Selezione del numero di bitplane

Valore	Numero di bitplane
000	Nessuno*
001	1
010	2
011	3
100	4
101	5
110	6**
111	Non utilizzato

* È visibile solo il colore di fondo.
** Il sesto bitplane viene usato soltanto nei modi dual-playfiel, HAM e EHB.

I bit nel registro BPLCON0 non sono modificabili singolarmente: per modificare un solo bit occorre riscriverli comunque tutti.

L'esempio seguente definisce un playfield in bassa risoluzione formato da due bitplane.

```
MOVE.W #$2200,BPLCON0+CUSTOM
```

Dal momento che il registro BPLCON0 influisce anche su altre caratteristiche del display, e dato che i bit non sono modificabili singolarmente, l'esempio precedente imposta anche altri parametri (su cui ci soffermeremo più avanti, in questo stesso capitolo). L'effetto complessivo è il seguente:

- Non viene usato il modo HAM.
- Viene usato il modo single-playfield.
- È abilitato il colore per l'uscita videocomposita (non applicabile a tutti i modelli).
- Viene disabilitato l'audio da genlock.
- La penna ottica viene disabilitata.
- Viene disattivato l'interlace.
- La sincronizzazione esterna tramite genlock viene disabilitata.

SELEZIONE DELLA RISOLUZIONE

Il normale televisore di casa è adatto in genere all'impiego in bassa risoluzione, con 320 pixel per linea. I monitor ad alta risoluzione monocromatici e RGB possono invece visualizzare schermi in alta risoluzione con 640 pixel per linea. Un oggetto disegnato in bassa risoluzione apparirà largo la metà se viene visualizzato in alta risoluzione.

Per selezionare l'alta risoluzione occorre impostare a 1 il bit 15, HIRES, di BPLCON0:

Alta risoluzione - impostare a 1 il bit 15.
Bassa risoluzione - impostare a 0 il bit 15.

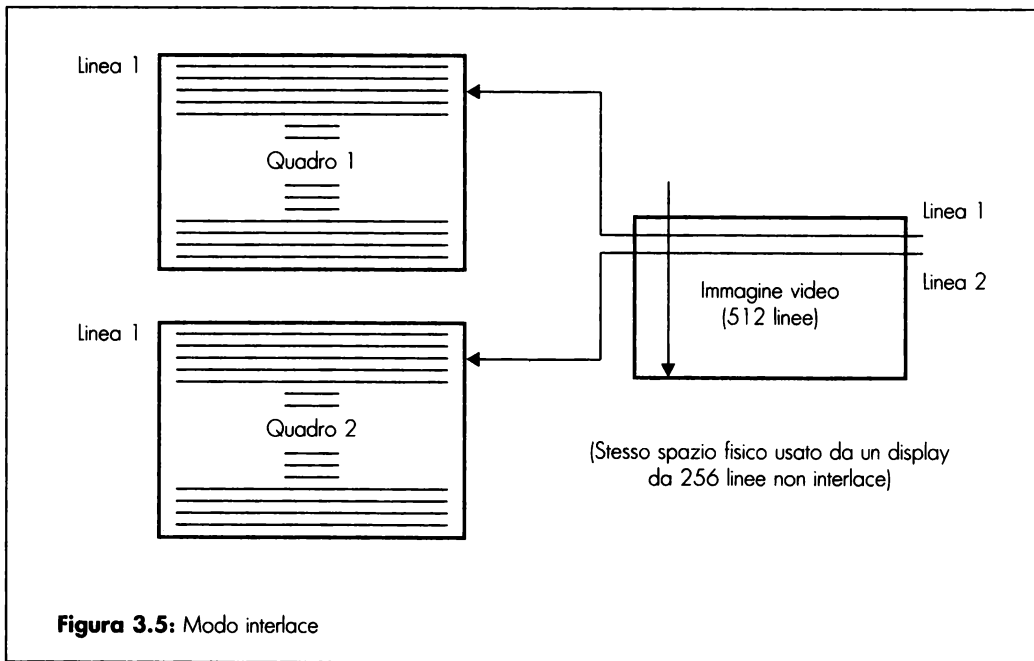
Si noti che in alta risoluzione è possibile avere un massimo di 4 bitplane per ogni playfield, e quindi un massimo di 16 colori.

Il modo interlace permette di raddoppiare la risoluzione verticale. La Tavola 3.6 mostra il numero di linee necessarie per riempire un normale schermo che non utilizzi l'overscan.

Tavola 3-6: Numero di linee in un normale playfield

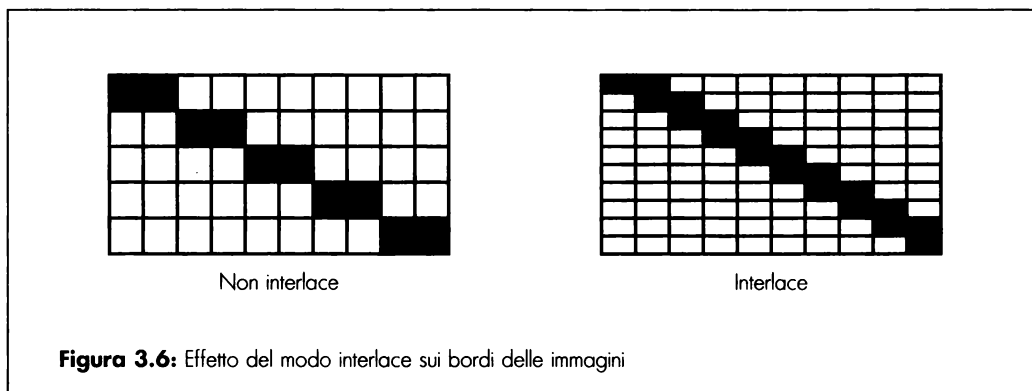
	PAL	NTSC
Non interlace	256	200
Interlace	512	400

In modo interlace, l'hardware di scansione fa sì che il secondo quadro video di ogni coppia

Figura 3.5: Modo interlace

sia spostato verso il basso di mezza linea di scansione.

Anche se il modo interlace richiede un po' di lavoro in più per predisporre i registri, come si vedrà in seguito, permette però di ottenere una precisione grafica altrimenti irraggiungibile. Si consideri l'aspetto della linea nella Figura 3.6 in modo interlace e non interlace. L'interlace elimina gran parte dell'"effetto scala" lungo la linea diagonale.

Se si utilizza il Blitter per tracciare linee in un playfield interlace, il playfield viene trattato come un unico display e non come due quadri video separati. Sono quindi sempre presenti i benefici effetti visti nella figura precedente.

Figura 3.6: Effetto del modo interlace sui bordi delle immagini

L'attivazione del modo interlace è controllata dal bit 2, LACE, di BPLCON0:

Modo interlace - Impostare a 1 il bit 2.
Modo non interlace - Impostare a 0 il bit 2.

Ancora una volta ricordiamo che i bit di BPLCON0 non sono modificabili singolarmente. L'esempio seguente definisce un playfield in alta risoluzione e in interlace.

```
MOVE.W #$A204,BPLCON0+CUSTOM
```

Dal momento che il registro BPLCON0 influisce anche su altre caratteristiche del display, e i bit non sono modificabili singolarmente, l'esempio precedente ha anche altri effetti.

- Viene abilitata l'alta risoluzione.
- Vengono usati due bitplane.
- Non viene usato il modo HAM.
- Viene usato il modo single-playfield.
- Viene abilitato il colore per l'uscita videocomposita.
- Viene disabilitato l'audio da genlock.
- La penna ottica viene disabilitata.
- Viene attivato il modo interlace.
- La sincronizzazione esterna tramite genlock viene disabilitata.

L'ammontare di memoria da allocare per ogni bitplane dipende dalla risoluzione, poiché display in alta risoluzione o in interlace richiedono bitplane di dimensioni maggiori.

L'ALLOCAZIONE DELLA MEMORIA PER I BITPLANE

Dopo aver scelto il numero di bitplane e aver determinato la risoluzione, viene il momento di allocare la memoria per le immagini. Ogni bitplane consiste di una serie di word in ordine d'indirizzo crescente. Se viene utilizzato il sistema operativo dell'Amiga è necessario fare uso della funzione AllocMem (o una equivalente) al fine di rimuovere un blocco di dimensioni appropriate dalla lista della memoria libera. Successivamente, i puntatori ai bitplane (BPLxPTH e BPLxPTL) vanno inizializzati in modo da puntare ai bitplane usati. L'indirizzo iniziale corrisponde all'angolo superiore sinistro del bitplane.

La Tavola 3.7 mostra la quantità di memoria necessaria per playfield normali. Può essere necessario cambiare la quantità di colori e la risoluzione a seconda della memoria disponibile.

Tavola 3.7: Memoria necessaria per un playfield (NTSC)

Dimensioni dell'immagine	Risoluzione	Numero di byte per bitplane
320 x 200	Bassa risoluzione non interlace	8.000
320 x 400	Bassa risoluzione interlace	16.000
640 x 200	Alta risoluzione non interlace	16.000
640 x 400	Alta risoluzione interlace	32.000

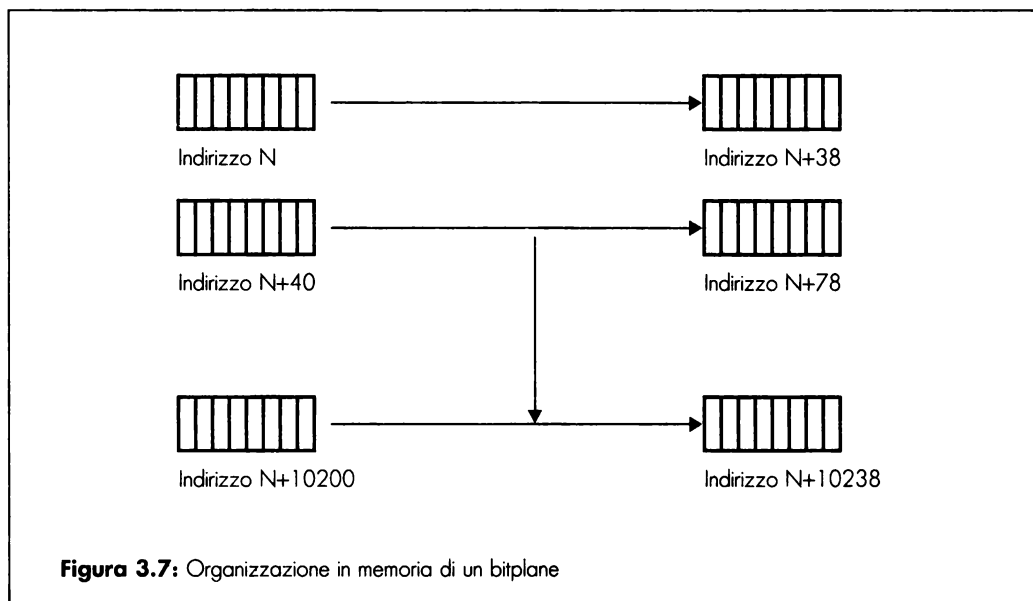
Tavola 3-8: Memoria necessaria per un playfield (PAL)

Dimensioni dell'immagine	Risoluzione	Numero di byte per bitplane
320 x 256	Bassa risoluzione non interlace	10.240
320 x 512	Bassa risoluzione interlace	20.480
640 x 256	Alta risoluzione non interlace	20.480
640 x 512	Alta risoluzione interlace	40.960

Per esempio, utilizzando un normale display PAL in bassa risoluzione non interlace con 256 linee di 320 pixel ciascuna, ogni linea del bitplane richiede 40 byte (320 bit = 40 byte). Moltiplicando le 256 linee per 40, si ottengono i 10.240 byte della tabella.

Un playfield in bassa risoluzione non interlace costituito da due bitplane richiede quindi un totale di 20.480 byte di memoria. La memoria allocata per un bitplane dev'essere continua, così occorrono due blocchi da 10.240 byte ciascuno. La figura 3.7 mostra un'area di 10.240 byte organizzata in 256 linee di 40 byte.

L'accesso alla memoria per i bitplane è controllato da due registri d'indirizzamento (BPLxPTH e BPLxPTL) per ogni bitplane, per un totale di 12 registri. La lettera "x" nel nome sta a indicare il numero del bitplane, per esempio, BPL1PTH e BPL1PTL contengono l'indirizzo iniziale del primo bitplane. Le coppie di registri i cui nomi terminano in PTH e PTL possono essere utilizzate

Figura 3.7: Organizzazione in memoria di un bitplane

dal 68000 come un registro unico delle dimensioni di una long word e iniziante all'indirizzo del registro che termina in PTH.

L'esempio seguente mostra come predisporre i puntatori ai bitplane. Supponendo che esistano due bitplane, uno all'indirizzo \$21000 e l'altro all'indirizzo \$25000, la CPU imposta BPL1PT a \$21000 e BPL2PT a \$25000. In genere questo compito viene lasciato al Copper.

```
;
;Dal momento che i registri appaiono come long word, possiamo
;utilizzare due istruzioni MOVE.L
;
      LEA    CUSTOM,a0          ;Indirizzo base dei chip custom
      MOVE.L $21000,BPL1PTH(a0) ;Puntatore al bitplane 1
      MOVE.L $25000,BPL2PTH(a0) ;Puntatore al bitplane 2
```

IMPOSTAZIONE DEI BITPLANE PER UNA CORRETTA COLORAZIONE

Dopo aver specificato il numero di bitplane e averne inizializzato i puntatori, si può cominciare a intervenire sui colori.

Un playfield da uno o due colori

Per un playfield monocromatico è necessario soltanto impostare a zero tutti i bit del bitplane, come si vede nell'esempio seguente. Il bitplane in questione viene riempito dal colore di sfondo (COLOR00) e tutta la sua area di memoria viene azzerata. Il bitplane inizia all'indirizzo \$21000 ed è lungo 10.240 byte.

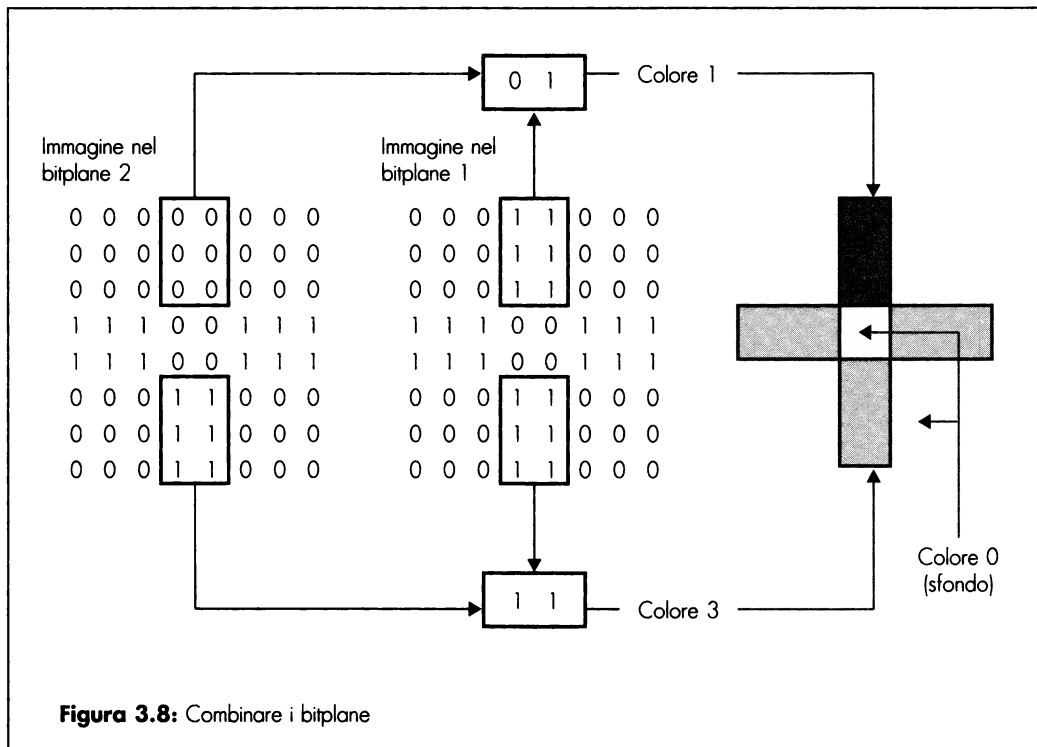
```
      LEA    $21000,a0  ;Punta al bitplane
      MOVE.W #2559,d0   ;2560 long word = 10240 byte
LOOP: MOVE.L #0,(a0)+   ;Azzera una longword
      DBRA   d0,LOOP    ;Ripete finche' d0 non e' minore di 0
```

Per un playfield a due colori occorre definire un bitplane che contenga 0 dove si desidera il colore di fondo e 1 dove si desidera il colore contenuto nel registro 1. L'esempio seguente è identico al primo, se non per il fatto che il bitplane viene riempito con il valore $FFOOFFOO anziché 0. Questo dà origine a due colori.

```
      LEA    $21000,a0          ;Punta al bitplane
      MOVE.W #2559,d0           ;2560 long word = 10240 byte
LOOP: MOVE.L #$FF00FF00,(a0)+   ;Imposta il valore $FF00FF00
      DBRA   d0,LOOP            ;Ripete finche' d0 non e' minore di 0
```

Un playfield di tre o più colori

Per tre o più colori, è necessario più di un bitplane. Ogni bitplane va impostato in modo che, quando i pixel sono combinati per la visualizzazione, venga prodotto il colore corretto. L'operazione è leggermente più complessa che con un solo bitplane. L'esempio che segue mostra un playfield a quattro colori, ma si può facilmente estendere a un numero qualsiasi di bitplane.

Figura 3.8: Combinare i bitplane

In un singolo playfield si possono combinare fino a cinque bitplane, in questo modo. L'uso di cinque bitplane permette una scelta di 32 colori per ogni pixel. La tabella al termine di questo capitolo riassume le combinazioni possibili per playfield formati da quattro o cinque bitplane.

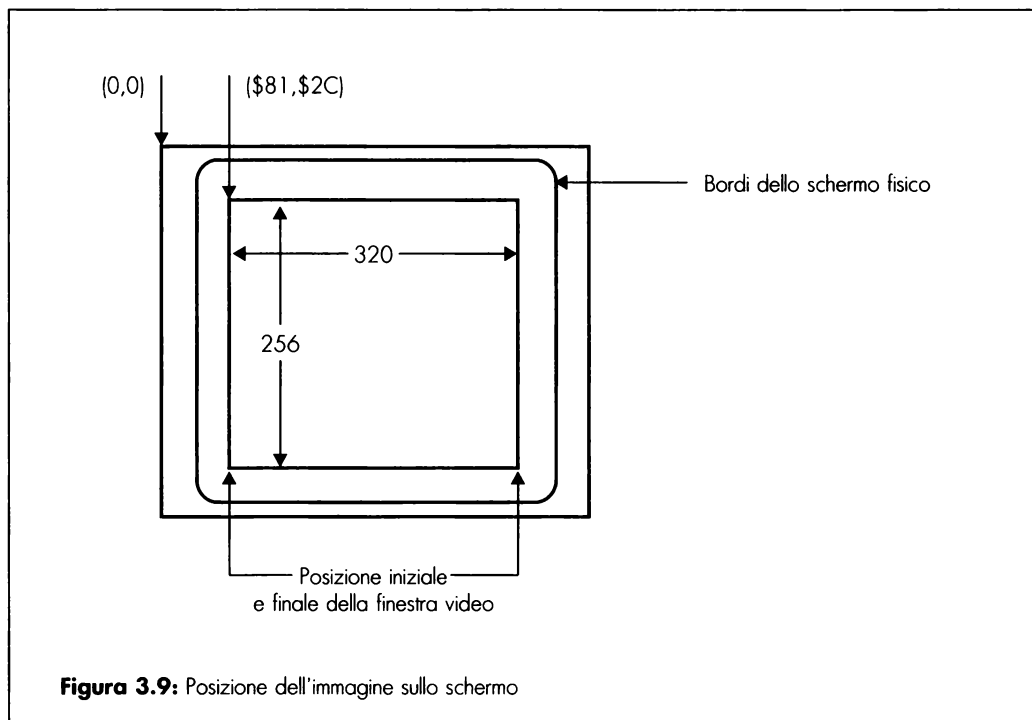
DEFINIZIONE DELLE DIMENSIONI DELLA FINESTRA VIDEO

Dopo aver completamente definito il playfield, occorre ora impostare le dimensioni della finestra video, ovverosia le dimensioni che avrà l'immagine sullo schermo. Quest'operazione non influisce soltanto sul playfield, ma sull'intera area video, incluso il bordo e gli sprite. Nessun oggetto può infatti essere visibile all'esterno della finestra video. Ovviamente, poi, anche le dimensioni del bordo esterno dipendono dalle dimensioni della finestra video.

Il semplice playfield descritto nel paragrafo precedente ha le stesse dimensioni dell'area di schermo utile e anche della finestra video. Ma questo non è obbligatorio: spesso la finestra video è più piccola dell'immagine in memoria (che si chiama raster). Una finestra video di questo tipo permette di visualizzare solo una parte del playfield, oppure di effettuarne lo scroll all'interno della finestra. È anche possibile creare finestre video di dimensioni maggiori di quelle normali. Tutti questi argomenti saranno trattati in maniera più estesa in seguito.

Le dimensioni della finestra video vengono definite tramite la sua posizione iniziale e quella finale, che andranno scritte negli appositi registri. La risoluzione verticale delle posizioni iniziale e finale è di una linea. La risoluzione orizzontale è di un pixel misurato in bassa risoluzione. Ogni punto dello schermo individua un pixel ben preciso tramite le sue coordinate x e y. Nel corso del libro indicheremo le coordinate x e y in questa forma: (x,y).

In genere la prima posizione dello schermo effettivamente utilizzata è ($81,$2C), anche se le coordinate iniziano dalla posizione (0,0) nell'angolo superiore sinistro dello schermo. Questo è

Figura 3.9: Posizione dell'immagine sullo schermo

vero sia per il sistema PAL che per quello NTSC.

L'hardware consente di specificare una posizione inferiore a ($81,$2C), ma parte dell'immagine potrebbe non risultare visibile su un normale monitor. Il pennello elettronico, infatti, si sposta su un'area molto più grande dello schermo principale per evitare possibili distorsioni alle estremità dell'immagine. La posizione ($81,$2C) è quella necessaria per centrare sullo schermo un'immagine di dimensioni normali, lasciando intorno un bordo di 8 pixel misurati in bassa risoluzione. La Figura 3.9 mostra le relazioni fra normale finestra video, area visibile sullo schermo, e area realmente coperta dal pennello elettronico.

Impostazione della posizione iniziale

Una posizione orizzontale di circa $81 e una verticale di circa $2C centrano l'immagine sulla maggior parte degli schermi televisivi. Selezionando l'alta risoluzione o il modo interlace, la posizione iniziale non cambia dal momento che viene sempre misurata in bassa risoluzione non interlace.

Il registro DIWSTRT controlla la posizione iniziale della finestra video. Questo registro contiene sia la posizione orizzontale (HSTART) sia quella verticale (VSTART). L'esempio seguente imposta DIWSTRT per un normale playfield. VSTART viene impostato a $2C e HSTART a $81.

```
LEA    CUSTOM,a0           ;Indirizzo base dei chip custom
MOVE.W #$2C81,DIWSTRT(a0)  ;Registro della posizione iniziale
```

Impostazione della posizione finale

È anche necessario comunicare al sistema la posizione finale della finestra video, che corrisponde al suo angolo inferiore destro. Essendo sempre interpretato in bassa risoluzione non interlace, anche questo valore non cambia se si seleziona l'alta risoluzione o il modo interlace.

Il registro DIWSTOP contiene sia il valore orizzontale (HSTOP) sia quello verticale (VSTOP) di questa posizione. Le seguenti istruzioni mostrano come impostare HSTOP e VSTOP per un normale playfield, presumendo una posizione iniziale ($81,$2C). Si noti che il valore di HSTOP corrisponde al valore reale meno 256 ($100). HSTOP è limitato alla metà destra dello schermo. Il normale valore di HSTOP, $1C1, viene quindi scritto come $C1. HSTOP ha il medesimo valore in PAL e in NTSC.

La posizione VSTOP è ristretta alla metà inferiore dello schermo. Ciò si ottiene via hardware, facendo sì che il bit più significativo sia il complemento a 1 di quello immediatamente successivo. Ciò permette una posizione di VSTOP superiore a 256 ($100) utilizzando solo 8 bit. La normale posizione finale in PAL è $2C (ovvero, per quanto appena detto, $12C) e $F4 in NTSC.

La posizione normale PAL di DIWSTRT è $2C81.
La posizione normale PAL di DIWSTOP è $2CC1.

La posizione normale NTSC di DIWSTRT è $2C81.
La posizione normale NTSC di DIWSTOP è $F4C1.

Nell'esempio seguente, il valore di DIWSTOP viene impostato con un HSTOP pari a $C1 e un VSTOP pari a $2C.

```
LEA    CUSTOM,a0          ;Base dei chip custom
MOVE.W $2CC1,DIWSTOP(a0) ;Registro della posizione finale
```

Tavola 3.9: Sommario di DIWSTRT e DIWSTOP

	Valori normali		Valori possibili	
	NTSC	PAL	Minimo	Massimo
DIWSTRT:				
VSTART	$2C	$2C	$00	$FF
HSTART	$81	$81	$00	$FF
DIWSTOP:				
VSTOP	$F4	$2C (= $12C)	$80	$7F (= $17F)
HSTOP	$C1	$C1	$00 (= $100)	$FF (= $1FF)

RECUPERO E VISUALIZZAZIONE DEI DATI

Dopo avere definito le dimensioni e la posizione della finestra video, è necessario assegnare la posizione in cui verranno visualizzati i dati provenienti dalla memoria. A tale scopo bisogna stabilire l'inizio e la fine di ogni riga e scriverne il valore negli appropriati registri. Questi registri hanno una risoluzione di 4 pixel, a differenza di quelli relativi alla finestra video che hanno una risoluzione di 1 pixel. Ogni posizione dista 4 pixel dalla precedente. Perciò la posizione 0 corrisponde al pixel 0, la posizione 1 al pixel 4 e così via.

La posizione iniziale dei dati e quella della finestra video dipendono l'una dall'altra. Il valore della posizione iniziale di lettura dei dati dovrebbe essere limitato a una risoluzione di 16 pixel (8 clock in bassa risoluzione e 4 in alta risoluzione). L'hardware richiede un certo intervallo tra la prima lettura dei dati e l'effettiva visualizzazione sullo schermo. Il risultato è una differenza di 4,5 clock di colore tra l'inizio della finestra video e la posizione iniziale di lettura dei dati.

Il normale valore di DDFSTRT in bassa risoluzione è $0038.
Il normale valore di DDFSTRT in alta risoluzione è $003C.

Si ricordi che la risoluzione relativa alla posizione iniziale e finale della finestra video è doppia di quella relativa al registro di lettura dei dati.

$$\frac{\$81}{2} - 8{,}5 = \$38$$

$$\frac{\$81}{2} - 4{,}5 = \$3C$$

La relazione tra la posizione iniziale di lettura dei dati e la posizione finale è la seguente:

DDFSTRT = DDFSTOP - (8 * (numero di word - 1)) in bassa risoluzione
DDFSTRT = DDFSTOP - (4 * (numero di word - 2)) in alta risoluzione

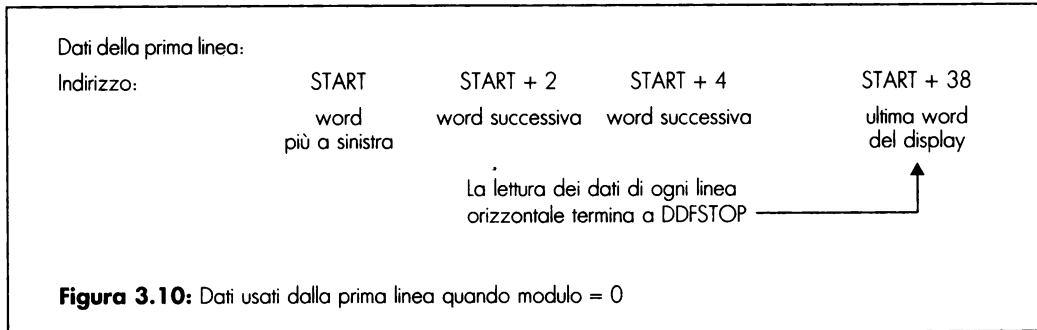
Il valore normale di DDFSTOP in bassa risoluzione è $00D0. Il valore normale in alta risoluzione è, invece, $00D4.

L'esempio seguente imposta DDFSTRT e DDFSTOP rispettivamente a $0038 e $00D0.

```
LEA    CUSTOM,a0            ;Base dei chip custom
MOVE.W #$0038,DDFSTRT(a0) ;Imposta DDFSTRT
MOVE.W #$00D0,DDFSTOP(a0) ;Imposta DDFSTOP
```

Il sistema deve anche sapere quali byte in memoria appartengono a ciascuna linea orizzontale del video. A questo scopo si ricorre al valore del modulo. Il "modulo" si riferisce al numero di byte che intercorrono tra l'ultima word appartenente a una linea orizzontale e la prima della successiva, permettendo quindi al sistema di trasformare l'organizzazione "sequenziale" dei dati in memoria in un'organizzazione "rettangolare". Per un normale playfield, le cui dimensioni sono uguali a quelle della finestra video, il modulo vale zero, dal momento l'area di memoria contiene esattamente il numero di byte che devono essere visualizzati. Le figure 3.10 e 3.11 mostrano esempi di bitplane in memoria.

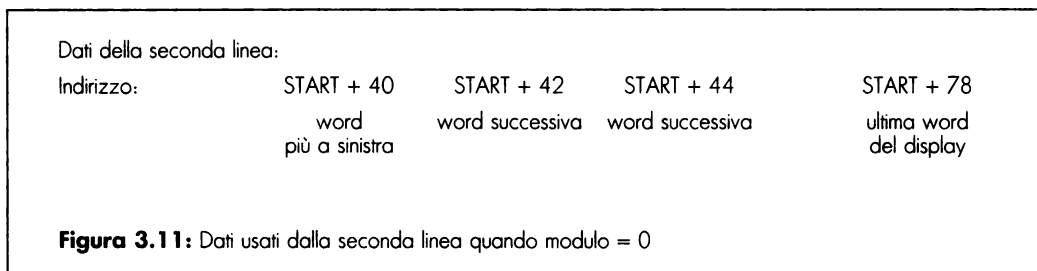
I puntatori ai bitplane (BPLxPTH e BPLxPTL) sono usati dal sistema per indirizzare i dati da trasferire sul video. Questi puntatori sono dinamici: una volta iniziato il trasferimento dei dati, vengono automaticamente incrementati in modo da puntare sempre alla word successiva (i dati vengono trasferiti una word alla volta). Quando viene raggiunta la condizione di fine linea, definita da DDFSTOP, ai puntatori viene aggiunto il valore del modulo perché puntino alla prima word della riga successiva.

Figura 3.10: Dati usati dalla prima linea quando modulo = 0

Al termine della prima riga, BPLxPTH e BPLxPTL contengono il valore START + 40. Il modulo (che qui vale 0) viene quindi sommato a questo valore in modo che la lettura dei dati per la riga successiva inizi al punto giusto. Nel caso attuale il valore è sempre START + 40.

Si noti che i puntatori contengono sempre un numero pari, perché i dati vengono trattati una

Dati della seconda linea:

Indirizzo:	START + 40	START + 42	START + 44	START + 78
	word più a sinistra	word successiva	word successiva	ultima word del display

Figura 3.11: Dati usati dalla seconda linea quando modulo = 0

word alla volta.

Vi sono due registri per il modulo: BPL1MOD per i bitplane dispari e BPL2MOD per quelli pari. Questo permette un modulo diverso per ognuno dei due playfield in modo dual-playfield. In genere, però, i due valori sono uguali.

Il seguente esempio imposta a zero il modulo per un playfield in bassa risoluzione formato da un solo bitplane. Il bitplane è considerato dispari.

```
MOVE.W   #0,BPL1MOD+CUSTOM   ;Imposta il modulo a 0
```

Trasferimento dei dati in alta risoluzione

Utilizzando l'alta risoluzione (in un playfield normale) si devono leggere 80 byte per riga anziché 40.

Il modulo in interlace

Usando il modo interlace, è necessario ridefinire il valore del modulo, poiché vengono generati due quadri separati per produrre l'immagine completa. Durante il primo passaggio vengono visualizzate le linee dispari, quelle pari nel secondo.

I bitplane di un normale schermo interlace sono formati da 512 linee anziché 256 (in PAL) o 400 anziché 200 (in NTSC). Supponendo che il playfield abbia una normale larghezza di 320 pixel, i dati dell'immagine in memoria avranno questa disposizione:

Linea 1 START
Linea 2 START + 40
Linea 3 START + 80
Linea 4 START + 120

e così via. Perciò occorre usare un modulo 40 per "saltare" le linee che appartengono all'altro quadro. Per i quadri dispari il puntatore al bitplane sarà inizializzato a START, per quelli pari a START + 40. Per queste operazioni si può usare il Copper.

VISUALIZZAZIONE DEL PLAYFIELD

Per inizializzare la visualizzazione dei dati occorre abilitare il canale DMA relativo ai bitplane, impostando a 1 il bit BPLEN del registro DMACON (si veda anche il Capitolo 7).

Inoltre, ogni volta che il playfield viene visualizzato si devono reimpostare i puntatori ai bitplane. Quest'operazione è necessaria in quanto i puntatori sono stati incrementati per puntare alla word successiva all'ultima della visualizzazione precedente e devono essere riportati al valore corretto. Il modo migliore di risolvere questo problema è tramite una lista di istruzioni del Copper, che si occupi dell'operazione durante l'intervallo di vertical blanking.

ABILITAZIONE DEL COLORE

Il modello A1000 possiede un'uscita videocomposita che viene abilitata impostando il bit 9 di BPLCON0. Senza l'aggiunta di appositi strumenti hardware, l'A500 e l'A2000 non possono generare un segnale di questo tipo.

L'abilitazione dell'uscita videocomposita non ha alcun effetto sul segnale RGB, che viene generato indipendentemente da questo segnale.

SOMMARIO DI UN SEMPLICE PLAYFIELD

I passi necessari per creare un playfield normale sono i seguenti.

1. Definizione delle caratteristiche del playfield

A. Altezza in linee.

- In PAL:
  * 256 in modo non interlace.
  * 512 in modo interlace.

- In NTSC:
  * 200 in modo non interlace.
  * 400 in modo interlace.

B. Larghezza in pixel.

- 320 in bassa risoluzione.

- 640 in alta risoluzione.

C. Colore dei pixel.

- Caricare i colori desiderati nei registri di colore.

- Definire il colore di ogni pixel tramite un valore binario che individui il registro corrispondente.

- Costruire i bitplane.

- Impostare i registri dei bitplane:

  * Bit 14-12 di BPLCON0: numero di bitplane (BPU2-BPU0).
  * BPLxPTH: puntatore al bitplane x (scritto come long word).

D. Risoluzione.

- Bassa risoluzione:

  * Impostare 320 pixel per linea.
  * Impostare a 0 il bit 15 di BPLCON0 (HIRES).

- Alta risoluzione:

  * Impostare 640 pixel per linea.
  * Impostare a 1 il bit 15 di BPLCON0 (HIRES).

E. Interlace.

- Modo interlace:

  * Impostare 512 linee in PAL, 400 in NTSC.
  * Impostare a 1 il bit 2 di BPLCON0 (LACE).

- Modo non interlace:

  * Impostare 256 linee in PAL, 200 in NTSC.
  * Impostare a 0 il bit 2 di BPLCON0 (LACE).

2. Allocazione della memoria.

Per calcolare i byte totali necessari per i bitplane si usa la seguente formula:

*Byte per linea * linee del playfield * numero di bitplane*

3. Definizione delle dimensioni della finestra video.

A. Posizione iniziale (da scrivere in DIWSTRT).

- I bit 15-8 rappresentano la posizione verticale.
- I bit 7-0 rappresentano la posizione orizzontale.

B. Posizione finale (da scrivere in DIWSTOP).

- I bit 15-8 rappresentano la posizione verticale.
- I bit 7-0 rappresentano la posizione orizzontale.

4. Definizione del trasferimento dei dati.

- Per DDFSTRT si utilizza la posizione orizzontale come abbiamo già spiegato.
- Per DDFSTOP si utilizza la posizione orizzontale come abbiamo già spiegato.

5. Definizione del modulo.

- Si imposrtano i registri BPL1MOD e BPL2MOD a 0 in modo non interlace e a 40 in modo interlace.

6. Preparazione della lista di istruzioni del Copper.

7. Abilitazione dell'uscita videocomposita.

- Solo per l'Amiga 1000: si imposta a 1 il bit 9 di BPLCON0. Ciò non influisce sull'uscita RGB.

ESEMPI DI CREAZIONE DI SEMPLICI PLAYFIELD

I seguenti esempi mostrano come impostare i registri e come scrivere la lista di istruzioni del Copper per due diversi playfield.

Il primo esempio riguarda un playfield 320 x 256 – costituito da un solo bitplane – che inizia alla locazione di memoria $21000. Vi è anche una lista di istruzioni del Copper all'indirizzo $20000.

Questo esempio si basa sul file include "hw_examples.i", contenuto nell'Appendice J.

```
      LEA     CUSTOM,a0            ;Base dei chip custom
      MOVE.W  #$1200,BPLCON0(a0) ;Un bitplane, uscita videocomposita abilitata
      MOVE.W  #0,BPLCON1(a0)     ;Scroll orizzontale = 0
      MOVE.W  #0,BPL1MOD(a0)     ;Modulo = 0
      MOVE.W  #$0038,DDFSTRT(a0) ;Inizio trasferimento dati a $38
      MOVE.W  #$00D0,DDFSTOP(a0) ;Termine trasferimento dati a $D0
      MOVE.W  #$2C81,DIWSTRT(a0) ;DIWSTRT = $2C81
      MOVE.W  #$2CC1,DIWSTOP(a0) ;DIWSTOP = $2CC1
      MOVE.W  #$0F00,COLOR00(a0) ;Colore di fondo = rosso
      MOVE.W  #$0FF0,COLOR01(a0) ;Colore 1 = giallo
;
;Riempie il bitplane di $FF00FF00 per produrre strisce verticali
;
      MOVEA.L #$21000,a1          ;Puntatore al bitplane
      MOVE.L  #$FF00FF00,d0       ;Dato da scrivere
      MOVE.W  #2559,d1            ;2560 long word = 10240 byte
LOOP: MOVE.L  d0,(a1)+            ;Scrive una long word
      DBRA    d1,LOOP             ;Ripete il loop
;
;Prepara la lista di istruzioni del Copper a $20000
;
      MOVEA.L #$20000,a1          ;Punta all'area destinazione
      LEA     COPPERL(pc),a2      ;Punta ai dati della lista
CLOOP:MOVE.L  a2,(a1)+            ;Muove una long word
      CMPI.L  #$FFFFFFFE,(a2)+    ;Controlla se e' il termine della lista
      BNE.S   CLOOP               ;In caso contrario ripete il loop
;
;Punta il Copper alla nuova lista
;
      MOVE.L  #$20000,COP1LCH(a0) ;Il registro di locazione
      MOVE.W  d0,COPJMP1(a0)      ;Ricarica il program counter
;
;Abilita il DMA
;
      MOVE.W  #(DMAF_SETCLR!DMAF_COPPER!DMAF_RASTER!DMAF_MASTER),DMACON(a0)
                                  ;Abilita il DMA dei bitplane e del Copper
      BRA     ....                ;Prosegue nel programma
;
;Seguono i dati della lista del Copper
;
      DC.W    BPL1PTH,$0002       ;Muove $0002 in $0E0 (BPL1PTH)
      DC.W    BPL1PTL,$1000       ;Muove $1000 in $0E2 (BPL1PTL)
```

```
	DC.W	$FFFF,$FFFE		;Fine della lista di istruzioni
```

Il secondo esempio riguarda invece un display in alta risoluzione interlace, sempre formato da un solo bitplane. Anche questo esempio necessita del file include "hw_examples.i".

```
	LEA	CUSTOM,a0		;Base dei chip custom
	MOVE.W	#$9204,BPLCON0(a0)	;Un bitplane, alta risoluzione interlace
	MOVE.W	#0,BPLCON1(a0)		;Scroll orizzontale = 0
	MOVE.W	#80,BPL1MOD(a0)		;Modulo = 80
	MOVE.W	#80,BPL2MOD(a0)		;Anche per i bitplane pari
	MOVE.W	#$003C,DDFSTRT(a0)	;Inizio trasferimento dati a $38
	MOVE.W	#$00D4,DDFSTOP(a0)	;Termine trasferimento dati a $D0
	MOVE.W	#$2C81,DIWSTRT(a0)	;DIWSTRT = $2C81
	MOVE.W	#$2CC1,DIWSTOP(a0)	;DIWSTOP = $2CC1
	MOVE.W	#$000F,COLOR00(a0)	;Colore di fondo = blu
	MOVE.W	#$0FFF,COLOR01(a0)	;Colore 1 = bianco
;
;Imposta un bitplane a $20000
;
	LEA	$20000,a1		;Puntatore al bitplane
	LEA	CHARLIST(pc),a2		;Puntatore ai dati
	MOVE.W	#511,d1			;512 linee di dati
	MOVE.W	#19,d0			;20 long word per linea
L1:
	MOVE.L	(a2),(a1)+		;Scrive una long word
	DBRA	d0,L1			;Riempie una linea
;
	MOVE.W	#19,d0			;Reimposta il contatore
	ADDQ.L	#4,a2			;Punta alla long word successiva
	CMPI.L	#$FFFFFFFF,(a2)		;Fine della lista?
	BNE.S	L2
	LEA	CHARLIST(pc),a2		;Ripristina il puntatore ai dati
L2:	DBRA	d1,L1			;Ripete per tutte le linee
;
;Abilita il DMA
;
	MOVE.W	#(DMAF_SETCLR!DMAF_RASTER!DMAF_MASTER),DMACON(a0)
					;Abilita il DMA dei bitplane

;Poiche' questo esempio non ha una lista di istruzioni
;per il Copper, si limita a entrare in attesa dell'intervallo
;di vertical blanking.
;Quando questo si verifica, controlla, tramite il bit LOF
;del registro VPOSR, se il quadro video seguente riguarda le linee
;dispari (quadro lungo) o quelle pari (quadro corto).
;Se LOF = 0, si tratta di un quadro corto, e i puntatori
;ai bitplane vengono impostati a $20050. Se LOF = 1
;si tratta di un quadro lungo e i puntatori vengono impostati
;a $20000.
```

```
VLOOP:MOVE.W  INTREQR(a0),d0        ;Legge le richieste di interrupt
      AND.W   #$0020,d0             ;Maschera tutto eccetto VBLANK
      BEQ.S   VLOOP                 ;Se necessario ripete il loop
      MOVE.W  VPOSR(a0),d0          ;Bit 15 di d0 = LOF
      BPL.S   VL1                   ;Salta se LOF = 0
      MOVE.L  #$20000,BPL1PTH(a0)   ;BPL1PT = $20000
      BRA.S   VLOOP                 ;Ricomincia il loop
VL1:
      MOVE.L  #$20050,BPL1PTH(a0)   ;BPL1PT = $20050
      BRA.S   VLOOP                 ;Ricomincia il loop
;
;Dati dell'immagine
;
CHARLIST:
      DC.L    $18FC3DF0,$3C6666D8,$3C66C0CC,$667CC0CC
      DC.L    $7E66C0CC,$C36666D8,$C3FC3DF0,$00000000
      DC.L    $FFFFFFFF
```

Creazione di uno schermo dual-playfield

Per una maggiore flessibilità nell'impostazione dello schermo, è possibile specificare due playfield anziché uno. Nel modo dual-playfield, il secondo playfield viene visualizzato sopra il primo. Per esempio, l'immagine relativa a un gioco potrebbe essere costituita da un riquadro in cui si svolge l'azione e da un pannello di controllo in primo piano. È possibile in questo modo modificare ciascuna delle due immagini senza dover ridisegnare l'intero schermo ogni volta. I due playfield possono anche essere mossi indipendentemente l'uno dall'altro.

Rispetto a uno schermo a playfield singolo, lo schermo dual-playfield presenta le seguenti differenze:

- I due playfield sono formati da un massimo di tre bitplane.
- I colori di ogni playfield (fino a 7 più il trasparente) corrispondono a diversi registri di colore.
- È necessario impostare un particolare bit per attivare il modo dual-playfield.

La Figura 3.12 mostra un esempio di schermo dual-playfield.

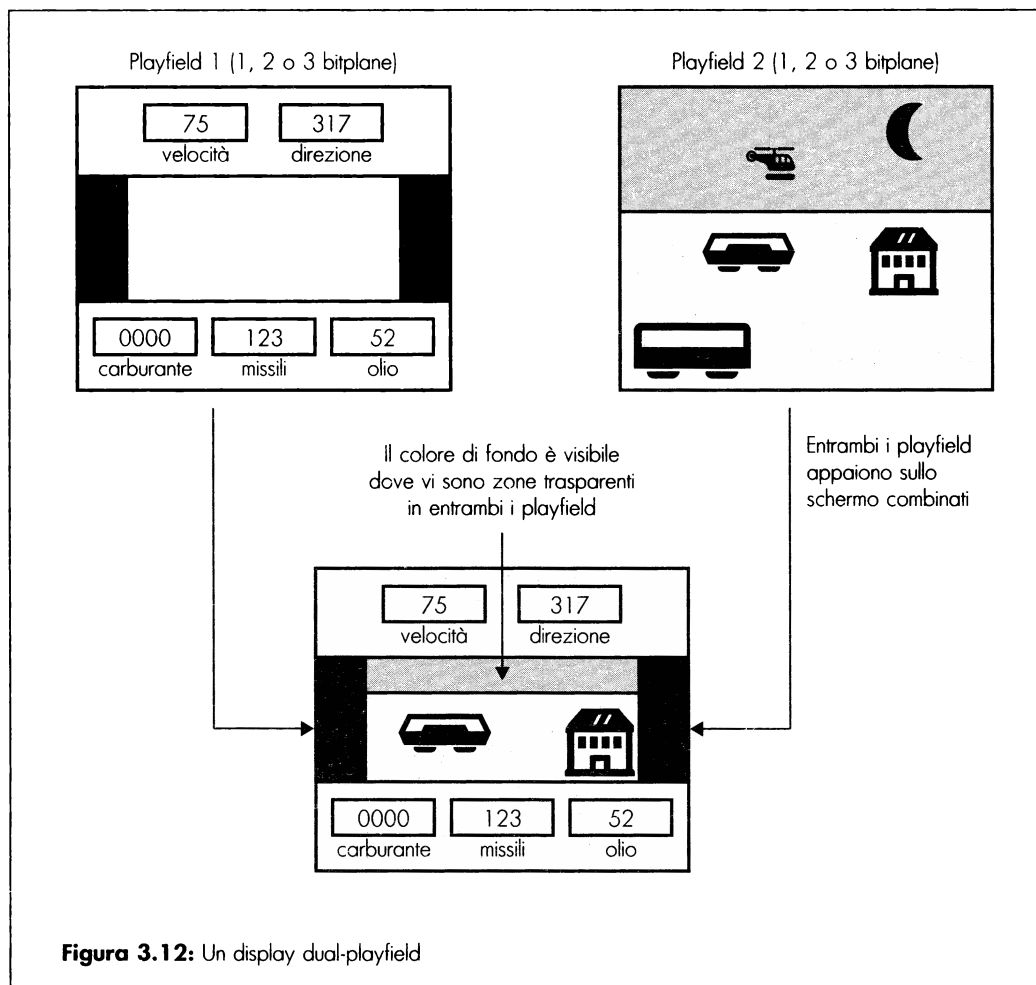
Nella figura, uno dei colori di ogni playfield è "trasparente" (il colore 0 nel playfield 1 e il colore 8 nel playfield 2). Questa caratteristica permette di lasciare in vista alcune parti dell'immagine sottostante.

Nel modo dual-playfield, ogni playfield è formato da un massimo di tre bitplane. I registri di colore da 0 a 7 sono assegnati al playfield 1 mentre quelli da 8 a 15 sono assegnati al playfield 2.

Assegnazione dei bitplane nel modo dual-playfield

I tre bitplane dispari (1, 3 e 5) sono raggruppati tra loro via hardware e vengono usati per il primo playfield. I tre bitplane pari (2, 4 e 6) vengono utilizzati invece per il secondo playfield. I bitplane sono assegnati ai playfield alternativamente, come mostra la Figura 3.13.

In alta risoluzione si possono avere al massimo due bitplane per ogni playfield. I bitplane 1 e 3 per il playfield 1 e i bitplane 2 e 4 per il playfield 2.

Figura 3.12: Un display dual-playfield

I REGISTRI DI COLORE NEL MODO DUAL-PLAYFIELD

Utilizzando il modo dual-playfield, l'hardware ricava i registri di colore per il playfield 1 dalla combinazione dei bit nei bitplane 1, 3 e 5. I bit provenienti dal bitplane 5 sono quelli più significativi. Quelli provenienti dal bitplane 1 sono i meno significativi. La Tavola 3.10 mostra come avviene questa selezione.

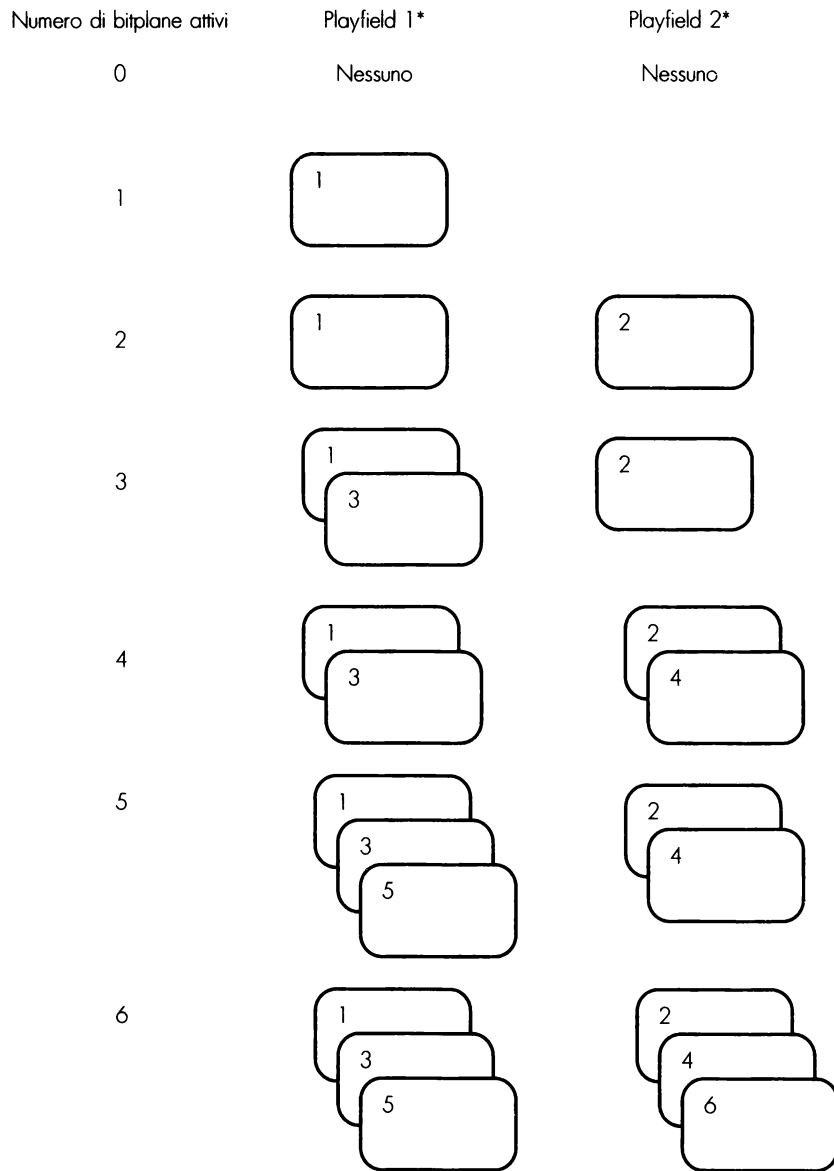



Figura 3.13: Come sono assegnati i bitplane nel modo dual-playfield

Tavola 3.10: Registri di colore del playfield 1 (bassa risoluzione)

PLAYFIELD 1

Combinazione di bit	Colore selezionato
000	Trasparente
001	COLOR01
010	COLOR02
011	COLOR03
100	COLOR04
101	COLOR05
110	COLOR06
111	COLOR07

Per quanto riguarda il playfield 2, l'hardware ricava i registri di colore dalla combinazione dei bit nei bitplane 2, 4 e 6. I bit provenienti dal bitplane 6 sono quelli più significativi. Quelli provenienti dal bitplane 2 sono i meno significativi. La Tavola 3.11 mostra come avviene questa selezione.

Tavola 3.11: Registri di colore del playfield 2 (bassa risoluzione)

PLAYFIELD 2

Combinazione di bit	Colore selezionato
000	Trasparente
001	COLOR09
010	COLOR10
011	COLOR11
100	COLOR12
101	COLOR13
110	COLOR14
111	COLOR15

La combinazione 000 seleziona il modo trasparente, lascia cioè vedere il colore di ciò che sta "dietro" al playfield (l'altro playfield, uno sprite o il colore di fondo).

La Tavola 3.12 mostra i registri di colore per il modo dual-playfield in alta risoluzione.

Tavola 3.12: Registri di colore dei playfield 1 e 2 (alta risoluzione)

PLAYFIELD 1

Combinazione di bit	Colore selezionato
00	Trasparente
01	COLOR01
10	COLOR02
11	COLOR03

PLAYFIELD 2

Combinazione di bit	**Colore selezionato**
00	Trasparente
01	COLOR09
10	COLOR10
11	COLOR11

CONTROLLO E PRIORITÀ NEL MODO DUAL-PLAYFIELD

Ognuno dei due playfield può avere priorità sull'altro, essere visualizzato, cioè davanti all'altro. In genere viene assegnata la priorità al playfield 1, comunque la scelta dipende dall'impostazione del bit 6, PF2PRI, del registro BPLCON2: quando è a 1, la priorità va al playfield 2, quando è a 0 la priorità va al playfield 1.
I due playfield hanno controlli separati per quanto riguarda:

- La dimensione in memoria e la parte dell'immagine visualizzata sullo schermo.

- Il movimento dell'immagine sullo schermo.

È necessario fare molta attenzione quando si muove un playfield sullo schermo mantenendo l'altro stazionario. Nello scroll dei playfield in bassa risoluzione, infatti, è necessario trasferire una word di dati in più della larghezza del playfield che si desidera muovere (due word in più in alta risoluzione). Esistono però soltanto due registri per controllare la posizione iniziale e finale del trasferimento dei dati (DDFSTRT e DDFSTOP), e quindi i due registri servono per entrambi i playfield. Volendone muovere uno è necessario intervenire su questi registri, e modificare il modulo e i puntatori ai bitplane relativi al playfield che non viene spostato, per mantenere inalterata la sua posizione sullo schermo. In bassa risoluzione i puntatori e il modulo vanno diminuiti di 2. In alta risoluzione di 4.

ATTIVAZIONE DEL MODO DUAL-PLAYFIELD

Per attivare il modo dual-playfield, è sufficiente impostare a 1 il bit 10, DBLPF, del registro BPLCON0. Questo modifica il modo in cui l'hardware raggruppa i bitplane per la determinazione del colore (bitplane pari e bitplane dispari vengono divisi in due gruppi) e modifica anche il modo in cui l'hardware muove i bitplane sullo schermo.

SOMMARIO DEL MODO DUAL-PLAYFIELD

I passi necessari per la creazione di uno schermo dual-playfield sono analoghi a quelli relativi a un playfield normale. Il procedimento presenta soltanto le seguenti differenze:

- **Caricamento dei registri di colore.** Si ricordi che i registri 0-7 sono usati dal playfield 1 e quelli 8-15 dal playfield 2.

- **Preparazione dei bitplane.** Il playfield 1 è formato dai bitplane 1, 3 e 5. Il playfield 2 dai bitplane 2, 4 e 6.

- **Valore del modulo.** BPL1MOD contiene il modulo relativo al playfield 1. BPL2MOD quello relativo al playfield 2.

Inoltre, sono necessari i seguenti passi:

- **Definizione della priorità.** Se si desidera che il playfield 2 abbia priorità sul playfield 1 occorre impostare a 1 il bit 6, PF2PRI, di BPLCON2.

- **Attivazione del modo dual-playfield.** Si deve impostare a 1 il bit 10, DBLPF, di BPLCON0.

Bitplane e finestre video di diverse misure

Si è visto finora come creare playfield grandi quanto la finestra video. Il paragrafo seguente illustra invece la procedura per la gestione di playfield la cui immagine in memoria sia più grande della finestra video, per creare finestre video più piccole o più grandi del normale e per lo spostamento della finestra video all'interno di un playfield di dimensioni maggiori.

QUANDO L'IMMAGINE IN MEMORIA È PIÙ GRANDE DELLA FINESTRA VIDEO

Se l'immagine in memoria è più grande della finestra video, è necessario decidere quale parte dell'immagine dev'essere visualizzata in quest'ultima. Questa procedura si diversifica da quella di gestione di un normale playfield per i seguenti aspetti:

- Il modulo ha valore diverso da zero.

- È necessaria una maggiore quantità di memoria per l'immagine.

Determinazione del modulo

Se l'immagine in memoria è più larga della finestra video, è necessario impostare un valore del modulo diverso da zero. Per esempio, supponiamo che la finestra video sia larga 320 pixel: per ogni linea saranno quindi visualizzati 40 byte di dati. Supponiamo però che l'immagine in memoria sia larga il doppio, cioè 80 byte, e supponiamo ancora di voler visualizzare la metà sinistra dell'immagine. La Figura 3.14 mostra la relazione tra l'immagine in memoria e quella che si desidera visualizzare.

Dal momento che devono essere trasferiti 40 byte per linea, per la prima linea si verifica la situazione della Figura 3.15.

A questo punto, BPLxPTH e BPLxPTL contengono l'indirizzo START + 40. Il modulo, impostato a 40, viene aggiunto a questo valore in modo che la linea successiva mostri i dati corretti. Il risultato si vede nella Figura 3.16.

Volendo visualizzare invece la metà destra dell'immagine, è necessario modificare il valore iniziale dei puntatori ai bitplane a START + 40 invece che a START, ma il modulo rimane sempre 40. Questa situazione è illustrata dalle figure 3.17 e 3.18.

Ora i puntatori ai bitplane contengono il valore START + 80. L'aggiunta del modulo (40) fa sì che i dati relativi alla seconda linea vengano trasferiti correttamente.

Si rammenti che in alta risoluzione viene utilizzata una quantità doppia di dati. Per un display normale sono necessari 80 byte anziché 40.

Figura 3.14: Immagine in memoria più grande del video

Figura 3.15: Dati usati dalla prima linea quando modulo = 40

Dati della linea 2:

Indirizzo:	START + 80	START + 82	START + 84	START + 118
	word più a sinistra	word successiva	word successiva	ultima word del display

Figura 3.16: Dati usati dalla seconda linea quando modulo = 40

Dati della linea 1:

Indirizzo:	START + 40	START + 42	START + 44	START + 78
	word più a sinistra	word successiva	word successiva	ultima word del display

Figura 3.17: Formato della prima linea – Metà destra dell'immagine

Dati della linea 2:				
Indirizzo:	START + 120	START + 122	START + 124	START + 158
	word più a sinistra	word successiva	word successiva	ultima word del display

Figura 3.18: Formato della seconda linea – Metà destra dell'immagine

Il trasferimento dei dati

I registri DDFSTRT e DDFSTOP specificano rispettivamente la posizione iniziale e finale dei dati trasferiti dalla memoria per ogni linea dello schermo. Il loro uso in questi casi è analogo a quello descritto nei paragrafi precedenti.

Allocazione della memoria

Per un'immagine più grande occorre ovviamente allocare una maggior quantità di memoria. La seguente formula generale permette di calcolarne l'ammontare preciso:

*byte per linea * numero di linee * numero di bitplane*

Perciò, se il playfield trattato nell'esempio precedente fosse stato costituito da due bitplane, avrebbe richiesto:

*80 * 256 * 2 = 40960 byte*

Posizione iniziale della finestra video

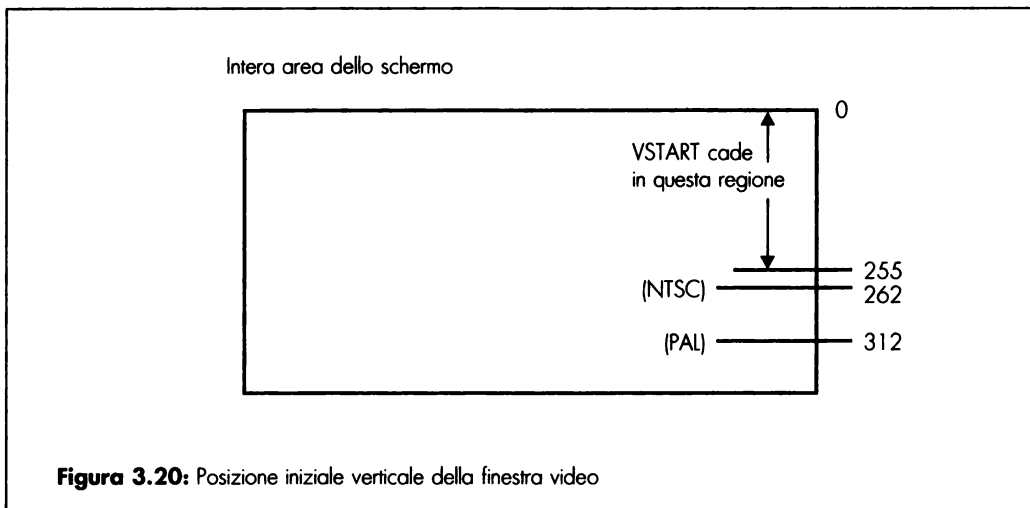
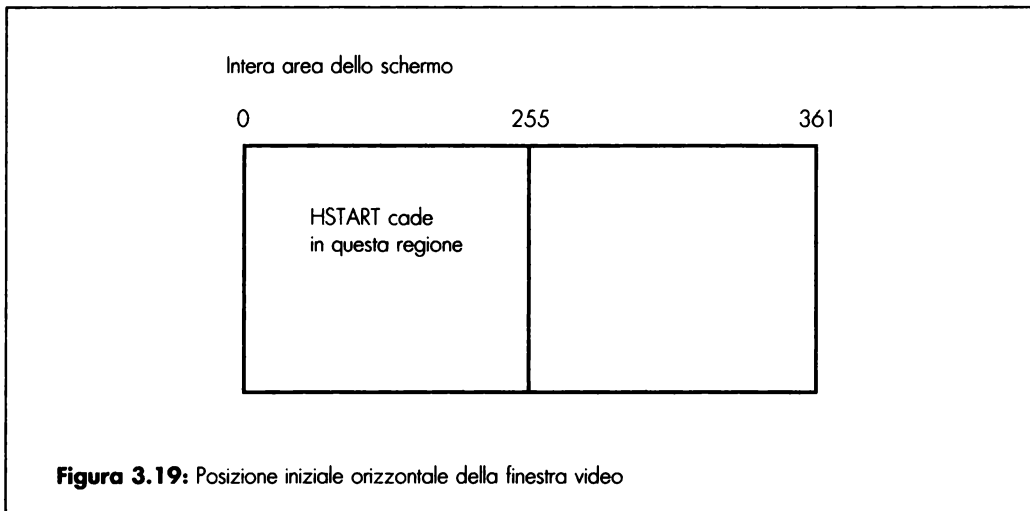
La posizione iniziale della finestra video è determinata dalle coordinate del suo angolo superiore sinistro. Il registro DIWSTRT contiene entrambe le componenti, orizzontale e verticale, di questa posizione, rispettivamente HSTART e VSTART. Gli otto bit relativi a HSTART corrispondono alle prime 256 posizioni possibili sullo schermo, a partire dall'estrema sinistra. La finestra video può iniziare in una qualsiasi di queste posizioni.

Gli otto bit relativi a VSTART corrispondono alle prime 256 posizioni possibili a partire dalla cima dello schermo.

Si tenga presente che questi valori sono sempre misurati come se il display fosse in bassa risoluzione non interlace.

Per impostare la posizione iniziale della finestra si scrive il valore di HSTART nei bit 0-7 e quello di VSTART nei bit 8-15 di DIWSTRT.

Figura 3.19: Posizione iniziale orizzontale della finestra video

Figura 3.20: Posizione iniziale verticale della finestra video

Posizione finale della finestra video

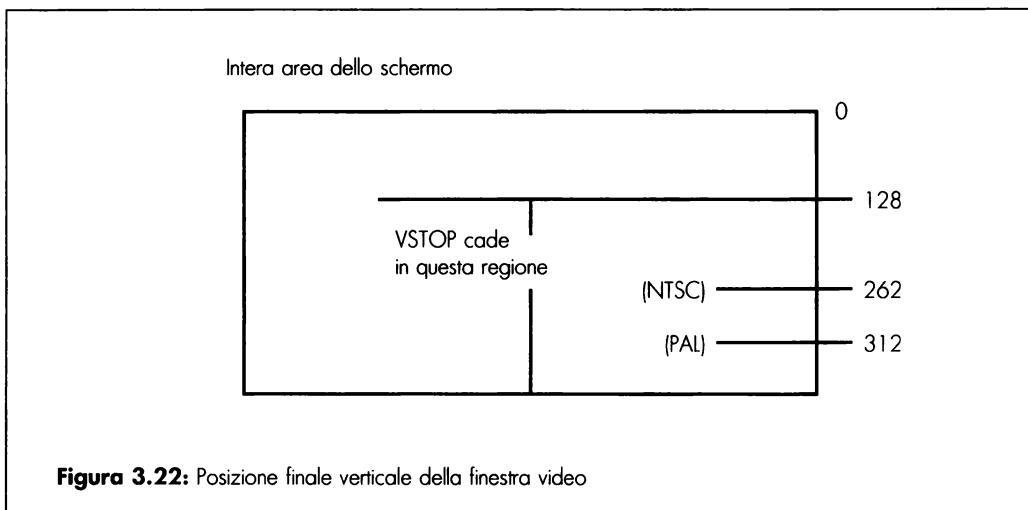
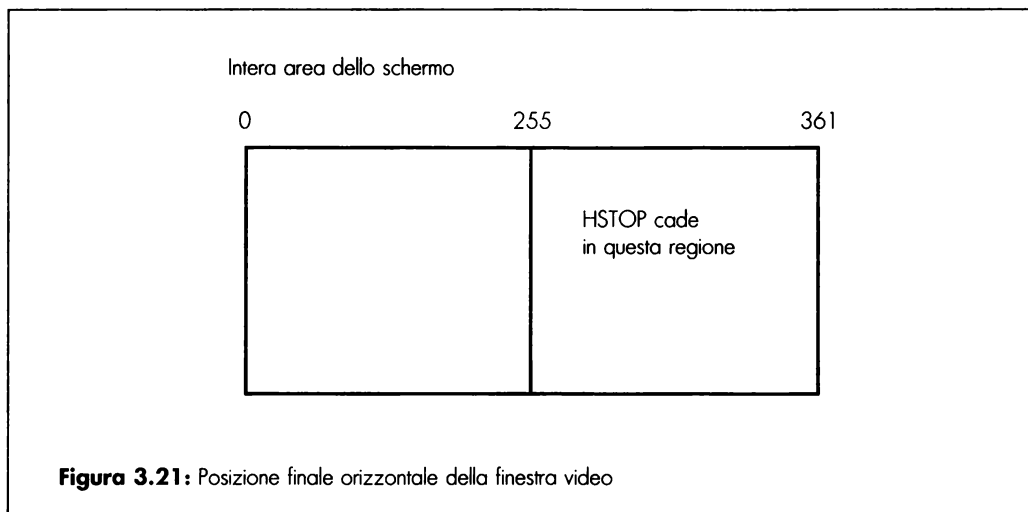
La posizione finale della finestra video è determinata dalle coordinate del suo angolo inferiore destro. Il registro DIWSTOP contiene entrambe le componenti, orizzontale e verticale, di questa posizione, rispettivamente HSTOP e VSTOP.

Per quanto riguarda l'impostazione di questo registro si veda anche il paragrafo "Creazione di un playfield".

Si ricordi che anche questo valore viene interpretato come se il display fosse in bassa risoluzione non interlace.

Per impostare la posizione finale della finestra video si scrive il valore di HSTOP nei bit 0-7 e quello di VSTOP nei bit 8-15 di DIWSTOP.

Figura 3.21: Posizione finale orizzontale della finestra video

Figura 3.22: Posizione finale verticale della finestra video

DIMENSIONI MASSIME DELLA FINESTRA VIDEO

Verticalmente, i limiti imposti alla finestra video sono determinati dal sistema impiegato (PAL o NTSC), che regola il massimo numero di linee e la durata dell'intervallo di vertical blanking durante il quale non possono essere visualizzati dati di nessun genere. La situazione è riassunta dalla tavola seguente.

Tavola 3.13: Dimensioni massime dello schermo (in verticale)

vertical blanking	PAL		NTSC	
Inizio	0		0	
Fine	$1D (29)		$15 (21)	
	PAL normale	PAL interlace	NTSC normale	NTSC interlace
Numero di linee disponibili	283	567 = 625 - (29 * 2)	241	483 = 525 - (21 * 2)

La situazione orizzontale è simile. L'hardware pone un limite minimo di $18 a DDFSTRT e di $D8 a DDFSTOP. Ciò significa un massimo di 25 word in bassa risoluzione e di 49 in alta risoluzione, poiché il limite massimo rimane a $D8 e in questa posizione può essere letta una sola word. In pratica, però, l'intervallo di horizontal blanking limita la quantità di dati visualizzabile a 368 pixel in bassa risoluzione (23 word). Questi valori sono uguali in PAL e in NTSC. Si tenga presente, inoltre, che impostare il registro DDFSTRT a un valore inferiore a $38 disabilita alcuni sprite.

Tavola 3.14: Dimensioni massime dello schermo (in orizzontale)

	Bassa risoluzione	Alta risoluzione
DDFSTRT (standard)	$0038	$003C
DDFSTOP (standard)	$00D0	$00D4
DDFSTRT (minimo)	$0018	$0018
DDFSTOP (massimo)	$00D8	$00D8
max numero di word per linea	25	49
max numero di pixel per linea	368 (bassa risoluzione)	

Movimento dei playfield

Volendo ottenere un effetto di movimento dello sfondo, è sufficiente costruire un playfield più grande dello schermo e muoverlo all'interno di quest'ultimo. Se si utilizza il modo dual-playfield, uno dei due playfield può essere mosso indipendentemente dall'altro.

Per ottenere un effetto di scroll verticale, è sufficiente aumentare o diminuire di una quantità pari alle dimensioni di una linea i valori contenuti nei puntatori ai bitplane. Il risultato è che l'immagine "si sposta" di una linea verso l'alto o verso il basso.

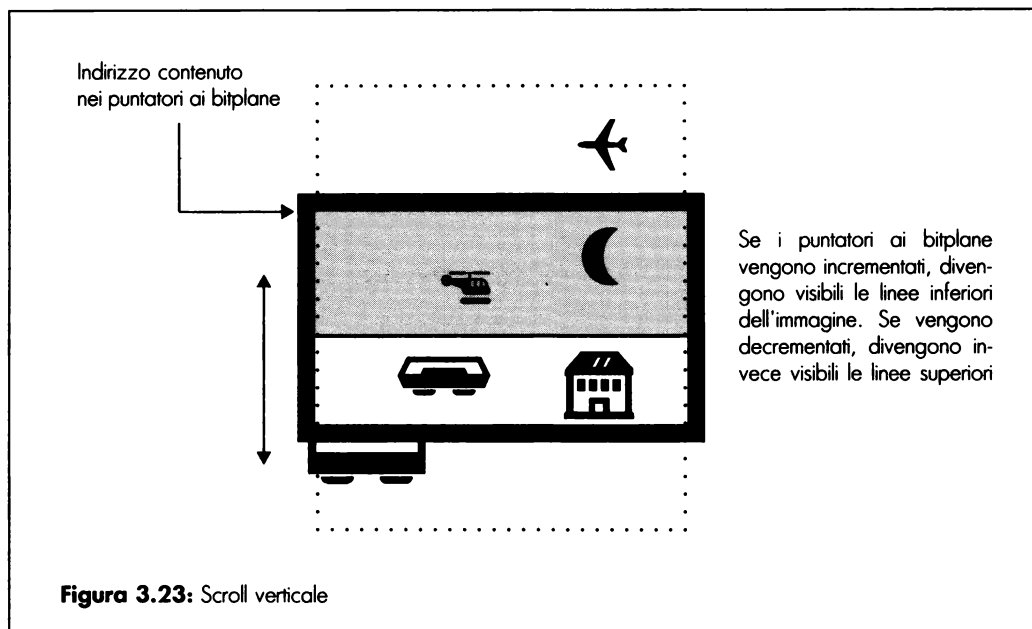
Lo scroll orizzontale funziona in modo diverso: è necessario predisporre il trasferimento di una word di dati in più per ogni riga e ritardarne quindi la visualizzazione in modo opportuno.

Entrambi i metodi si prestano bene per essere controllati tramite le liste di istruzioni del Copper. In tal modo, la corretta impostazione dei registri avviene automaticamente durante l'intervallo di vertical blanking.

SCROLL VERTICALE

È possibile muovere un playfield in su o in giù nella finestra video. A ogni successiva visualizzazione, i puntatori ai bitplane corrispondono a un indirizzo progressivamente più alto o più basso nell'immagine in memoria. Se i puntatori vengono aumentati, appaiono sullo schermo più dati relativi alla parte inferiore dell'immagine, che sembrerà quindi muoversi verso l'alto. Viceversa, diminuendo il valore dei puntatori, diventa visibile la parte più alta dell'immagine, che si sposterà così verso il basso.

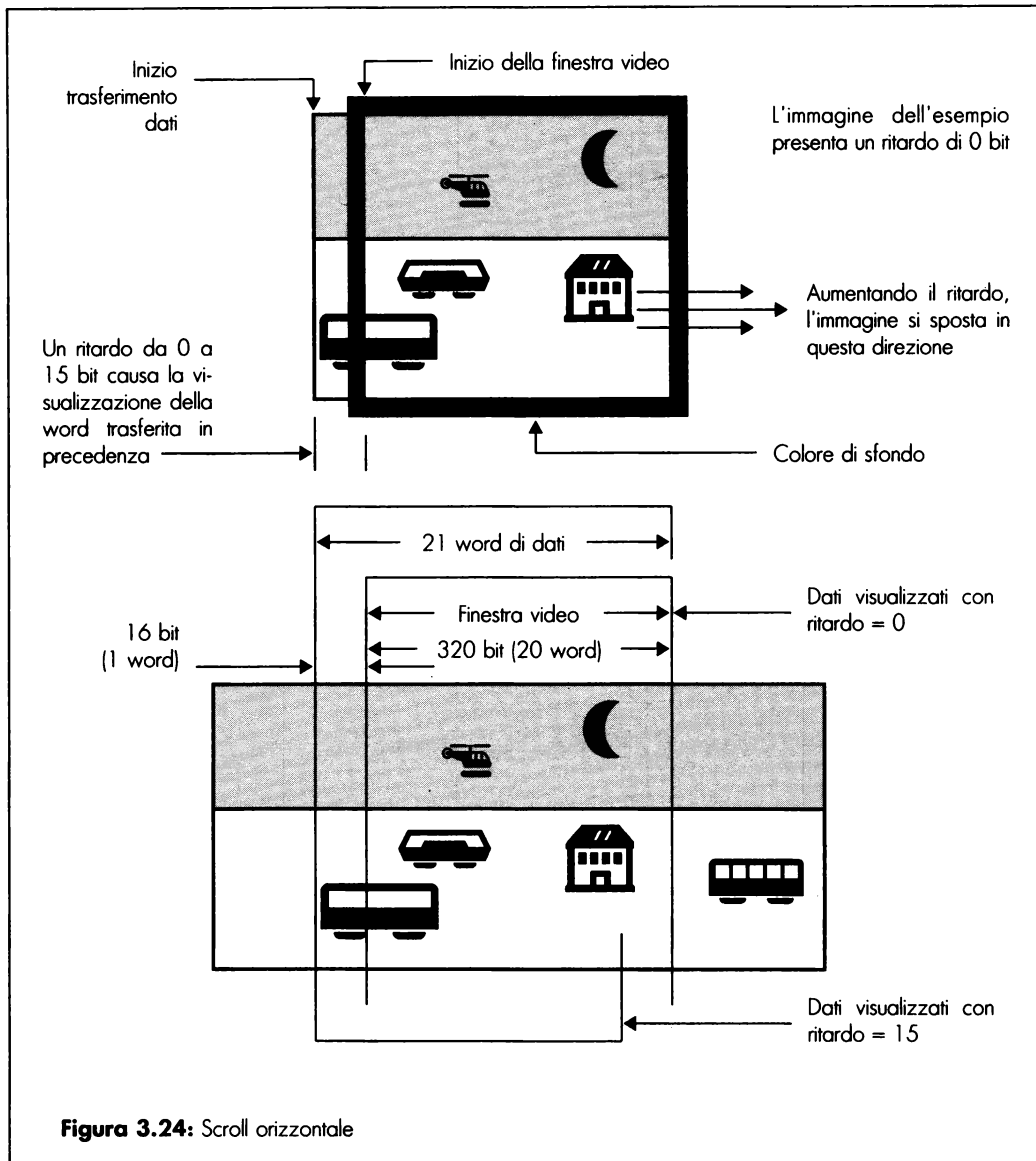
In un sistema PAL, con un display di 256 linee, ogni movimento può essere quindi pari a 1/256 dell'altezza totale. In modo interlace questo valore può giungere a 1/512 dello schermo se si modificano correttamente tutti i registri coinvolti nell'operazione. Si raccomanda comunque di effettuare lo scroll di due linee per volta, al fine di mantenere le linee del playfield nella corretta relazione pari/dispari. In un sistema NTSC, questi valori diventano rispettivamente 1/200 e 1/400 dell'altezza totale.

Figura 3.23: Scroll verticale

Per predisporre un playfield allo scroll verticale, è necessario creare bitplane sufficientemente alti da permettere lo scroll desiderato, calcolare i nuovi valori dei puntatori e far sì che il Copper li utilizzi nella successiva visualizzazione dei dati.

Si supponga, per esempio, di voler muovere un playfield verso l'alto di una linea per volta. Prima della visualizzazione di ogni quadro sarà quindi necessario incrementare i puntatori in modo che puntino alla linea successiva. Il che, per un normale playfield in bassa risoluzione con modulo 0, si traduce in un aumento di 40 byte per volta.

SCROLL ORIZZONTALE

I playfield possono anche essere mossi orizzontalmente. La velocità di questo scroll viene controllata impostando un "valore di ritardo". La parola "ritardo" sta a indicare che una word di dati in più viene trasferita sullo schermo, ma la sua visualizzazione viene "ritardata" per un opportuno intervallo. La word viene piazzata all'estrema sinistra dello schermo prima del normale punto d'inizio del trasferimento dati. A ogni spostamento dell'immagine verso destra, i bit di questa word diventano visibili all'estrema sinistra mentre quelli all'estrema destra scompaiono dal video. Per ogni pixel di ritardo, i dati sullo schermo si spostano di un pixel verso destra. Maggiore è il ritardo, maggiore è la velocità di scroll. Si possono impostare fino a 15 pixel

Figura 3.24: Scroll orizzontale

di ritardo. In alta risoluzione ogni pixel di ritardo corrisponde a uno spostamento di due pixel. La Figura 3.24 illustra in che modo il ritardo e il trasferimento di una word supplementare provocano l'effetto di scroll.

Per predisporre un playfield allo scroll orizzontale, è necessario:

- Definire bitplane sufficientemente larghi da permetterlo.
- Impostare i registri di trasferimento dati perché leggano una word in più per ogni linea.
- Impostare i bit di ritardo.
- Impostare correttamente il modulo e i puntatori ai bitplane.
- Scrivere le istruzioni del Copper per controllare i cambiamenti durante l'intervallo di vertical blanking.

Trasferimento dei dati per lo scroll orizzontale

Il normale inizio del trasferimento dati per un playfield statico è $38. Volendo ottenere uno scroll orizzontale, il trasferimento deve iniziare con una word di anticipo (DDFSTRT = $0030). Tra l'altro, ciò disabilita lo sprite numero 7. DDFSTOP non viene modificato. Si rammenti inoltre che nel modo dual-playfield, l'impostazione dei registri influisce su entrambi i playfield.

Impostazione del modulo nello scroll orizzontale

Come sempre, il valore del modulo corrisponde alla differenza tra gli indirizzi della prima word di una riga e dell'ultima della riga precedente meno 2. Si consideri per esempio un display di 40 byte all'interno di un'immagine larga 80 byte: dal momento che lo scroll orizzontale richiede il trasferimento di due byte in più, verranno letti 42 byte per ogni linea.

Ora i puntatori ai bitplane contengono il valore START + 42. Se si imposta quindi il modulo a 38, si ottiene il valore corretto per la linea successiva.

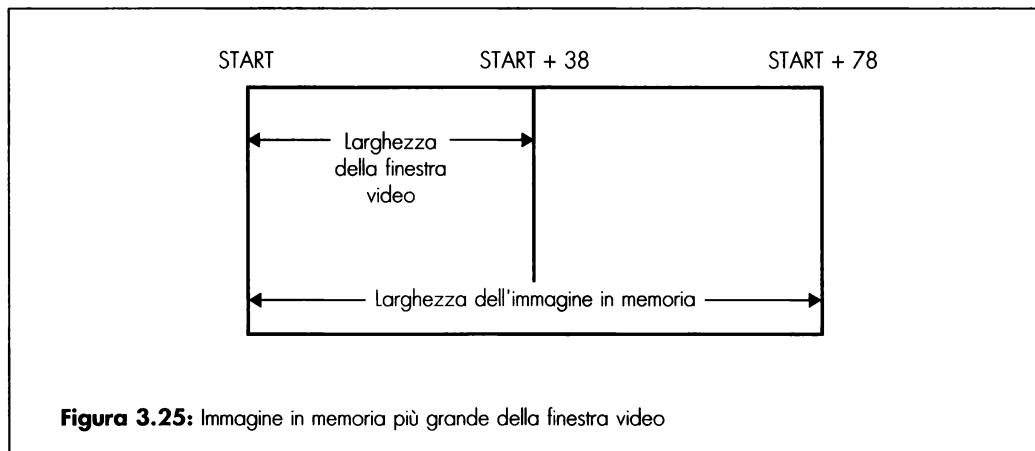
Figura 3.25: Immagine in memoria più grande della finestra video

Dati della linea 1:				
Indirizzo:	START	START + 2	START + 4	START + 40
	word più a sinistra	word successiva	word successiva	ultima word del display

Figura 3.26: Scroll orizzontale – dati della prima linea

Dati della linea 2:				
Indirizzo:	START + 80	START + 82	START + 84	START + 120
	word più a sinistra	word successiva	word successiva	ultima word del display

Figura 3.27: Scroll orizzontale – dati della seconda linea

Valore del ritardo

Il valore del ritardo nello scroll orizzontale è controllato dai bit 7-0 di BPLCON1. Ogni playfield è controllabile separatamente: i bit 3-0 corrispondono al playfield 1 (bitplane 1, 3 e 5), mentre i bit 7-4 corrispondono al playfield 2 (bitplane 2, 4 e 6)

Conviene impostare sempre tutti gli otto bit, anche se si utilizza un solo playfield. In questo caso, i valori dei bit 3-0 e 7-4 saranno uguali.

L'esempio seguente imposta il ritardo a 7 per entrambi i playfield.

```
MOVE.W  #$77,BPLCON1+CUSTOM
```

SOMMARIO DELLO SCROLL DEI PLAYFIELD

I passi necessari per creare un playfield in grado di spostarsi sullo schermo sono quasi uguali a quelli relativi a un playfield normale. Il procedimento presenta soltanto le seguenti differenze:

- **Impostazione del trasferimento dati.** Si deve trasferire una word in più per ogni linea, iniziando 16 pixel più a sinistra.

- **Valore del modulo.** Il modulo è inferiore di 2 rispetto al normale.

Inoltre, sono necessari i seguenti passi:

- **Scroll verticale.** Si devono modificare in modo opportuno i puntatori ai bitplane, impostando BPLxPTH e BPLxPTL durante l'intervallo di vertical blanking.

- **Scroll orizzontale.** Si deve specificare il ritardo, impostando i bit 7-0 di BPLCON1 per un massimo di 15 bit di ritardo.

Informazioni avanzate

Questo paragrafo riguarda caratteristiche usate meno frequentemente od opzionali.

INTERAZIONI TRA I PLAYFIELD E GLI ALTRI OGGETTI VIDEO

I playfield condividono lo schermo con gli sprite. Il Capitolo 7 mostra in che modo si assegnano le priorità video ai playfield e agli sprite e come si rilevano le collisioni (sovrapposizioni) tra questi elementi.

MODO HAM (HOLD AND MODIFY)

Il modo HAM è uno speciale modo video che consente di ottenere 4096 colori diversi contemporaneamente. In genere il colore di un pixel dipende dal valore contenuto nel relativo registro (ovvero dalla combinazione dei bit dei bitplane). Il modo HAM, invece, prevede che il colore generato per l'ultimo pixel venga mantenuto (hold) anche per il pixel seguente, eventualmente modificandone (modify) una delle componenti (rossa, verde o blu).

Il modo HAM permette di ottenere sfumature di colore e ombreggiature molto graduali. Un sistema è quello di disegnare gli oggetti usando i normali colori della palette, e poi, tramite il modo HAM, aggiungere ombreggiature o colori completamente nuovi. Si noti però che ogni pixel può presentare modifiche in una sola delle tre componenti del colore. L'effetto risulta quindi abbastanza limitato.

Nel modo HAM si fa uso di tutti e sei i bitplane disponibili. I bitplane 5 e 6 vengono sfruttati per modificare il criterio con cui sono considerati i bitplane 1-4. Le convenzioni sono le seguenti:

- Se la combinazione dei bit 6-5 per un dato pixel è 00, viene usata la normale procedura di selezione del colore (cioè i bit 4-1 vengono usati per selezionare uno dei primi 16 registri di colore). Se si utilizzano solo cinque bitplane, i dati relativi al sesto sono automaticamente considerati nulli.

- Se la combinazione dei bit 6-5 è 01, il pixel a sinistra di quello considerato assume un colore identico nei livelli del rosso e del verde, e diverso nel livello del blu: i bit nei bitplane 4-1 vengono usati per rimpiazzare i quattro bit relativi al livello del blu.

- Se la combinazione dei bit 6-5 è 10, il pixel a sinistra di quello considerato assume un colore identico nei livelli del blu e del verde, e diverso nel livello del rosso: i bit nei bitplane 4-1 vengono usati per rimpiazzare i quattro bit relativi al livello del rosso.

- Se la combinazione dei bit 6-5 è 11, il pixel a sinistra di quello considerato assume un colore identico nei livelli del rosso e del blu, e diverso nel livello del verde: i bit nei bitplane 4-1 vengono usati per rimpiazzare i quattro bit relativi al livello del verde.

Usando il modo HAM è possibile limitarsi a definire un solo registro di colore per lo sfondo (COLOR00). L'intero schermo viene poi trattato modificando questo colore di base, come si è visto nello schema precedente.

Il bit 11 del registro BPLCON0 è quello che attiva il modo HAM, ma è necessario intervenire anche sui seguenti bit di BPLCON0:

- Il bit HOMOD, il bit 11, dev'essere impostato a 1.

- Il bit DBLPF, il bit 10, dev'essere impostato a 0 (modo dual-playfield disattivato).
- Il bit HIRES, il bit 15, dev'essere impostato a 0 (bassa risoluzione).
- I bit BPU2-BPU0, i bit 14-12, devono essere impostati a 101 o 110 (5 o 6 bitplane attivi).

Il seguente programma di esempio genera un playfield in modo HAM formato da 6 bitplane. Tutti e 32 i registri di colore sono impostati con il colore nero per provare che i colori sono generati dal modo HAM.

```
;
;Prima i registri di controllo.
;
      LEA     CUSTOM,a0          ;Base dei chip custom
      MOVE.W  #$6A00,BPLCON0(a0) ;6 bitplane, HAM
      MOVE.W  #0,BPLCON1(a0)     ;Scroll = 0
      MOVE.W  #0,BPL1MOD(a0)     ;Modulo dei bitplane dispari = 0
      MOVE.W  #0,BPL2MOD(a0)     ;Modulo dei bitplane pari = 0
      MOVE.W  #$0038,DDFSTRT(a0) ;Inizio trasferimento dati
      MOVE.W  #$00D0,DDFSTOP(a0) ;Fine trasferimento dati
      MOVE.W  #$2C81,DIWSTRT(a0) ;Inizio finestra video
      MOVE.W  #$F4C1,DIWSTOP(a0) ;Fine finestra video
;
;Imposta con il colore nero tutti i registri per dimostrare
;che i colori sono generati dal modo HAM.
;
      MOVE.W  #31,d0             ;Inizializza il contatore
      LEA     CUSTOM+COLOR00,a1  ;Punta ai registri di colore
CLOOP:MOVE.W  #$0000,(a1)+       ;Imposta un registro
      DBRA    d0,CLOOP           ;Ripete il ciclo
;
;Riempie i bitplane con una matrice facilmente riconoscibile
;
      MOVE.W  #1999,d0           ;duemila long word per bitplane
      MOVE.W  #$21000,a1         ;a1 punta al bitplane 1
      MOVE.W  #$23000,a2         ;a2 punta al bitplane 2
      MOVE.W  #$25000,a3         ;a3 punta al bitplane 3
      MOVE.W  #$27000,a4         ;a4 punta al bitplane 4
      MOVE.W  #$29000,a5         ;a5 punta al bitplane 5
      MOVE.W  #$2B000,a6         ;a6 punta al bitplane 6
FLOOP:
      MOVE.L  #$55555555,(a1)+
      MOVE.L  #$33333333,(a2)+
      MOVE.L  #$0F0F0F0F,(a3)+
      MOVE.L  #$00FF00FF,(a4)+
      MOVE.L  #$CF3CF3CF,(a5)+
      MOVE.L  #$3CF3CF3C,(a6)+
      DBRA    d0,FLOOP           ;Ripete il loop
;
;Prepara una lista del Copper a $20000
```

```
        MOVE.L  #$20000,a1           ;Indirizzo della lista
        LEA     COPPERL(pc),a2       ;Indirizzo dei dati della lista
COPLOOP:
        MOVE.L  (a2),(a1)+           ;Trasferisce una long word
        CMPI.L  #$FFFFFFFE,(a2)+     ;Termine della lista?
        BNE.S   COPLOOP              ;Ripete il loop
        MOVE.L  #$20000,COP1LCH(a0)  ;Prepara i puntatori
        MOVE.W  d0,COPJMP1(a0)       ;Forza l'esecuzione
;
;Abilita il DMA.
;
        MOVE.W  #$8380,DMACON(a0)    ;Abilita bitplane e Copper

        BRA     ...resto del programma...
;
;Dati della lista del Copper
;
COPPERL:
        DC.W    BPL1PTH,$0002        ;Bitplane 1 a $21000
        DC.W    BPL1PTL,$1000
        DC.W    BPL2PTH,$0002        ;Bitplane 2 a $23000
        DC.W    BPL2PTL,$3000
        DC.W    BPL3PTH,$0002        ;Bitplane 3 a $25000
        DC.W    BPL3PTL,$5000
        DC.W    BPL4PTH,$0002        ;Bitplane 4 a $27000
        DC.W    BPL4PTL,$7000
        DC.W    BPL5PTH,$0002        ;Bitplane 5 a $29000
        DC.W    BPL5PTL,$9000
        DC.W    BPL6PTH,$0002        ;Bitplane 6 a $2B000
        DC.W    BPL6PTL,$B000
        DC.W    $FFFF,$FFFE          ;Fine della lista
```

CREAZIONE DI UNO SCHERMO CONTENENTE DIVERSI PLAYFIELD

La libreria grafica (graphics.library) contiene le funzioni di supporto necessarie per la divisione dello schermo in diverse viewport, ognuna con la propria risoluzione e palette di colori. Si vedano gli altri manuali della serie ROM Kernel per le informazioni relative.

USO DI UNA SORGENTE VIDEO ESTERNA

Per l'Amiga esistono apparecchiature chiamate "genlock", che permettono di sovrapporre allo schermo del computer immagini provenienti da sorgenti esterne come videoregistratori, videocamere o lettori di dischi laser. Utilizzando un'interfaccia genlock, l'immagine proveniente dalla sorgente esterna sostituisce il colore di fondo (il colore 0).

SOMMARIO DEI REGISTRI RIGUARDANTI I PLAYFIELD

Questo paragrafo riassume schematicamente tutti i registri citati in questo capitolo e il significato dei loro bit. Si consulti l'Appendice A per un elenco completo dei registri.

BPLCON0 – Controllo dei bitplane

I bit di questo registro non possono essere modificati indipendentemente l'uno dall'altro.

Bit 0 – non utilizzato

Bit 1 – ERSY
1 = sincronizzazione esterna abilitata (genlock)
0 = sincronizzazione esterna disabilitata

Bit 2 – LACE
1 = modo interlace abilitato
0 = modo non interlace abilitato

Bit 3 – LPEN
1 = penna ottica abilitata
0 = penna ottica disabilitata

Bit 4-7 non utilizzati (azzerare)

Bit 8 – GAUD
1 = audio del genlock abilitato
0 = audio del genlock disabilitato

Bit 9 – COLOR_ON
1 = abilitazione dell'uscita videocomposita
0 = disabilitazione dell'uscita videocomposita

Bit 10 – DBLPF
1 = selezione del modo dual-playfield
0 = playfield singolo

Bit 11 – HOMOD
1 = modo HAM attivo
0 = modo HAM inattivo

Bit 14, 13, 12 – BPU2, BPU1, BPU0
Numero di bitplane utilizzati

000 = solo colore di fondo
001 = 1 bitplane
010 = 2 bitplane
011 = 3 bitplane
100 = 4 bitplane

101 = 5 bitplane
110 = 6 bitplane
111 non utilizzato

Bit 15 - HIRES
1 = alta risoluzione
0 = bassa risoluzione

BPLCON1 - Controllo dei bitplane

Bit 3-0 - PF1H(3-0)
Ritardo playfield 1

Bit 7-4 - PF2H(3-0)
Ritardo playfield 2

Bit 15-8 non utilizzati

BPLCON2 - Controllo dei bitplane

Bit 5-0
Controllo priorità tra sprite e playfield

Bit 6 - PF2PRI
1 = priorità al playfield 2
0 = priorità al playfield 1

Bit 7-15 non utilizzati

DDFSTRT - Inizio trasferimento dati

Bit 15-8 non utilizzati

Bit 7-2
Posizione H8-H3. H3 usato solo in alta risoluzione

Bit 1-0 non utilizzati

DDFSTOP - Fine trasferimento dati

Bit 15-8 non utilizzati

Bit 7-2
Posizione H8-H3. H3 usato solo in alta risoluzione

Bit 1-0 non utilizzati

BPLxPTH - Puntatori ai bitplane

Word più significativa dei puntatori, x è il numero del bitplane

BPLxPTL – Puntatori ai bitplane

Word meno significativa dei puntatori, x è il numero del bitplane

DIWSTRT – Inizio della finestra video

Bit 15-8 – VSTART (V7-V0)

Bit 7-0 – HSTART (H7-H0)

DIWSTOP – Fine della finestra video

Bit 15-8 – VSTOP (V7-V0)

Bit 7-0 – HSTOP (H7-H0)

BPL1MOD – Modulo dei bitplane

Bitplane dispari, playfield 1

BPL2MOD – Modulo dei bitplane

Bitplane pari, playfield 2

Sommario della selezione dei colori

Questo paragrafo presenta un riepilogo delle procedure riguardanti la selezione dei colori nei playfield, comprese informazioni sul contenuto dei registri, esempi di colori e note sulle differenze che si riscontrano in alta e in bassa risoluzione.

CONTENUTO DEI REGISTRI DI COLORE

La Tavola 3.15 mostra il contenuto dei registri di colore, che sono tutti a sola scrittura.

Tavola 3.15: Contenuto dei registri di colore

Bit	Contenuto
15-12	Non utilizzati: azzerare
11-8	Rosso
7-4	Verde
3-0	Blu

ALCUNI ESEMPI DI COLORI

La Tavola 3.16 mostra alcuni esempi di colori ottenibili e i corrispondenti valori esadecimali.

Tavola 3.16: Valori dei registri e colori corrispondenti

Valore	Colore	Valore	Colore
$FFF	Bianco	$1FB	Acqua chiaro
$D00	Rosso mattone	$6FE	Blu cielo
$F00	Rosso	$6CE	Blu chiaro
$F80	Rosso-arancio	$00F	Blu
$F90	Arancione	$61F	Blu brillante
$FB0	Giallo-oro	$06D	Blu scuro
$FD0	Giallo-cadmio	$C1F	Violetto
$FF0	Limone	$FAC	Rosa
$8E0	Verde chiaro	$DB9	Beige
$0F0	Verde	$C80	Marrone
$2C0	Verde scuro	$A87	Marrone scuro
$0B1	Verde albero	$999	Grigio medio
$0DB	Acqua	$000	Nero

SELEZIONE DEI COLORI IN BASSA RISOLUZIONE

La Tavola 3.17 mostra come avviene la selezione dei colori in bassa risoluzione. Se la combinazione dei bit dai vari bitplane è quella mostrata, viene utilizzato il colore contenuto nel registro corrispondente.

Tavola 3.17: Colori in bassa risoluzione

Playfield singolo		Dual-playfield	
Modo normale	**Modo HAM**		**Registro usato**
		Playfield 1 Bitplane 5, 3, 1	
00000	0000	000	0*
00001	0001	001	1
00010	0010	010	2
00011	0011	011	3
00100	0100	100	4
00101	0101	101	5
00110	0110	110	6
00111	0111	111	7

		Playfield 2 Bitplane 6, 4, 2	
01000	1000	000**	8
01001	1001	001	9
01010	1010	010	10
01011	1011	011	11
01100	1100	100	12
01101	1101	101	13
01110	1110	110	14
01111	1111	111	15
10000			16
10001			17
10010			18
10011			19
10100	NON	NON	20
10101	UTILIZZATO	UTILIZZATO	21
10110	IN	IN	22
10111	QUESTO	QUESTO	23
11000	MODO	MODO	24
11001			25
11010			26
11011			27
11100			28
11101			29
11110			30
11111			31

* Il registro 0 definisce sempre il colore di fondo.
** Seleziona il modo trasparente, anziché il registro 8.

SELEZIONE DEI COLORI IN MODO HAM

In modo HAM, l'interpretazione dei bit nei bitplane segue regole diverse, illustrate dalla Tavola 3.18. Il modo HAM è attivo soltanto quando il bit 10 di BPLCON0 è impostato a 1.

Tavola 3.18: Selezione del colore in modo HAM

Bitplane 6	**Bitplane 5**	**Risultato**	
0	0	Operazione normale	Utilizza il registro corrispondente
0	1	Conserva verde e rosso	Blu = bitplane 4-1
1	0	Conserva verde e blu	Rosso = bitplane 4-1
1	1	Conserva rosso e blu	Verde = bitplane 4-1

SELEZIONE DEI COLORI IN ALTA RISOLUZIONE

La Tavola 3.19 mostra come avviene la selezione del colore in alta risoluzione.

Tavola 3.19: Colori in alta risoluzione

Playfield singolo Bitplane 4,3,2,1	Dual-playfield	Registro usato
	Playfield 1 Bitplane 3, 1	
0000	00*	0**
0001	01	1
0010	10	2
0011	11	3
0100		4
0101	NON UTILIZZATI	5
0110	IN QUESTO MODO	6
0111		7
	Playfield 2 Bitplane 4, 2	
1000	00*	8
1001	01	9
1010	10	10
1011	11	11
1100		12
1101	NON UTILIZZATI	13
1110	IN QUESTO MODO	14
1111		15

* Seleziona il modo trasparente.
** Il registro 0 definisce sempre il colore di fondo.

4 HARDWARE DEGLI SPRITE

Introduzione

Gli sprite sono oggetti hardware indipendenti dall'immagine presente sullo schermo e indipendenti anche l'uno dall'altro. Insieme ai playfield, gli sprite costituiscono l'immagine che appare sul monitor dell'Amiga. Gli sprite sono visualizzati sullo schermo tramite otto canali DMA riservati a questo scopo. Uno sprite è largo in genere 16 pixel e non ha limiti d'altezza. Ogni sprite ha a disposizione tre colori per la sua immagine, e può contenere anche pixel trasparenti, attraverso i quali si vedono tutte le immagini collocate sotto lo sprite. Per ottenere oggetti più grandi o più complessi, o per avere una più ampia scelta di colori, gli sprite possono essere combinati insieme.

I canali DMA relativi agli sprite possono essere riutilizzati più volte durante la scansione del video. Sullo schermo possono quindi comparire anche più di otto sprite contemporaneamente.

PRESENTAZIONE DEL CAPITOLO

In questo capitolo vengono trattati i seguenti argomenti:

- Come stabilire forma, dimensioni, colore e posizione di uno sprite.
- Come visualizzare e spostare gli sprite.
- Come combinare gli sprite per ottenere immagini più complesse, una larghezza maggiore o un maggior numero di colori.
- Come riutilizzare più volte il canale DMA di uno sprite per ottenere più di otto sprite sullo schermo nello stesso momento.

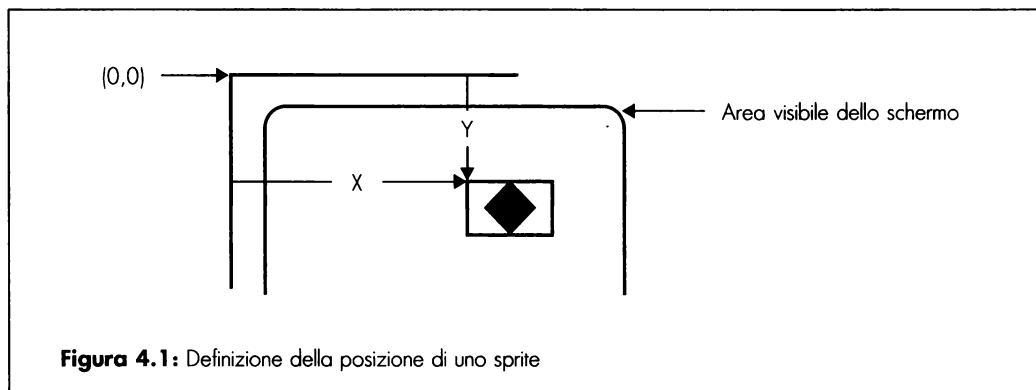
Creazione di uno sprite

Per poter utilizzare uno sprite, occorre prima definirne le caratteristiche principali in una particolare struttura di dati. I dati necessari sono i seguenti:

- La larghezza (fino a un massimo di 16 pixel).
- L'altezza (senza limitazioni).
- La forma.
- La scelta dei tre colori più il trasparente.
- La posizione sullo schermo.

POSIZIONE SULLO SCHERMO

La posizione di uno sprite è individuata da una coppia di coordinate X,Y. La coppia (0,0) corrisponde all'angolo in alto a sinistra del video. Le coordinate si riferiscono al pixel nell'angolo superiore sinistro dello sprite. Esse vanno sempre considerate come relative a uno schermo in bassa risoluzione non interlace. La Figura 4.1 mostra questa relazione. Si ricordi che, a causa dell'overscan video, la posizione (0,0) (cioè X=0, Y=0) cade al di fuori della regione visibile dello schermo.

Figura 4.1: Definizione della posizione di uno sprite

La grandezza dell'area utilizzabile dello schermo dipende anche dalle dimensioni della finestra video, definita tramite DDFSTRT, DDFSTOP, DIWSTRT e DIWSTOP. Per maggiori informazioni si veda il Capitolo 3.

Posizione orizzontale

La posizione orizzontale di uno sprite (la coordinata X) può corrispondere a qualunque pixel compreso tra 0 e 447. Tuttavia, per essere visibile, un oggetto deve trovarsi all'interno della finestra video. La zona di schermo normalmente utilizzabile va dal pixel 64 al 383 (per un totale

di 320 pixel). Il pennello elettronico spazia su un'area più larga, ma in genere non visibile sul monitor.

Se come posizione orizzontale viene specificato un valore esterno alla finestra video, tutto lo sprite, o parte di esso, potrebbe restare invisibile. A volte questo è proprio l'effetto desiderato; un simile sprite viene detto "clipped", tagliato.

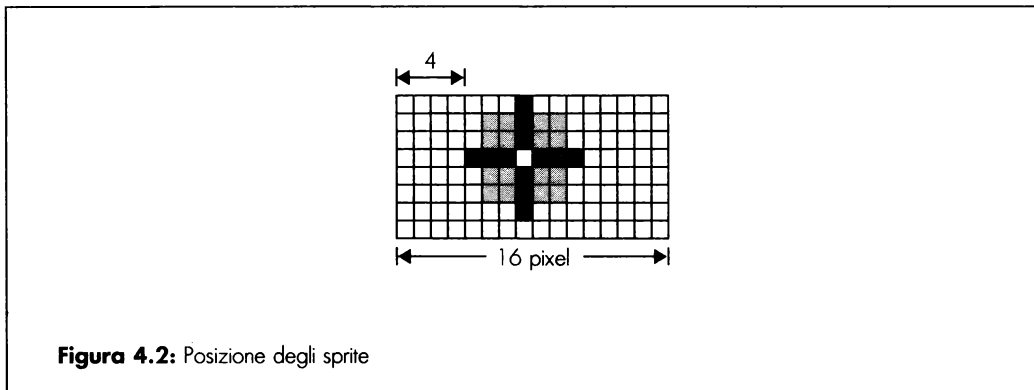
Per far sì che lo sprite appaia alla corretta posizione orizzontale, bisogna sommare l'offset sinistro della finestra video alla sua coordinata X. Nell'esempio della figura ciò significherebbe aggiungere 64 al valore di X. Per esempio, per far iniziare l'immagine a 94 pixel dal bordo sinistro dello schermo, è sufficiente eseguire questo calcolo:

Posizione desiderata + offset sinistro = 94 + 64 = 158

Perciò 158 è il valore che andrà inserito nella struttura dati.

Occorre inoltre tener presente che la posizione X rappresenta la posizione del pixel più a sinistra dello sprite. Questo anche se il pixel è trasparente e quindi non visibile sullo schermo.

Se lo sprite di Figura 4.2 fosse collocato nella posizione X = 158, l'immagine vera e propria apparirebbe 4 posizioni più a destra, al pixel 162. I primi quattro pixel, infatti, sono trasparenti, e non appaiono sullo schermo.

Figura 4.2: Posizione degli sprite

Posizione verticale

Il limite superiore dello sprite può essere collocato in qualunque posizione nell'intervallo 0-312 (PAL). Negli esempi di questo capitolo viene usata una finestra video PAL con coordinate y che vanno dalla linea 44 alla linea 299. Ciò da origine a un normale schermo di 256 linee in modo non interlace. Se viene specificata una posizione verticale (Y) minore di 44, la parte superiore dello sprite potrebbe non essere visibile.

In questo caso, quindi, per far sì che lo sprite appaia nella posizione desiderata basta sommare a questa il valore 44. Per esempio, affinché lo sprite venga visualizzato alla riga 25 dall'inizio della finestra video, il valore da inserire nella struttura dati sarà:

Posizione desiderata + offset verticale = 25 + 44 = 69

Sprite "tagliati"

Come abbiamo accennato in precedenza, gli sprite vengono totalmente o parzialmente tagliati quando oltrepassano i limiti dello schermo. I valori 64 (orizzontale) e 44 (verticale) sono "normali" per un'immagine centrata in un normale monitor PAL. Se vengono scelti altri valori per la finestra video, gli sprite vengono opportunamente tagliati.

DIMENSIONI DEGLI SPRITE

Gli sprite sono larghi 16 pixel e non hanno limiti di altezza: variano da un minimo di una linea a un massimo che supera l'altezza dello schermo (in quest'ultimo caso spesso si ricorre allo scroll per mostrare lo sprite un po' alla volta).

Le dimensioni di uno sprite sono basate sulla grandezza di un pixel in bassa risoluzione non interlace: 1/320 della larghezza normale e 1/256 dell'altezza. Gli sprite tuttavia sono indipendenti dalle caratteristiche dello schermo sottostante; in altre parole, cambiare la risoluzione orizzontale o verticale della finestra video *non ha alcun effetto* sulle dimensioni o sulla risoluzione di uno sprite.

FORMA DEGLI SPRITE

Uno sprite può avere qualunque forma, purché rispetti il limite dei 16 pixel orizzontali. Questa forma viene definita specificando quali pixel devono apparire in ciascuna posizione dello sprite. Ad esempio, le figure 4.3 e 4.4 mostrano un'astronave la cui forma è indicata dalle lettere "X". La prima figura mostra l'astronave in quello che potrebbe essere uno schizzo preliminare su carta. La seconda mostra invece l'astronave all'interno dei 16 pixel disponibili. Le lettere "O" tutt'intorno indicano le parti dello sprite che appariranno trasparenti sullo schermo.

Figura 4.3: Figura di un'astronave

```
0 0 0 0 X X 0 0 0 0 0 0 0 0 0 0
0 0 X X X X X X 0 0 0 0 0 0 0 0
X X X X X X X X X X 0 0 0 0 0 0
X X X X X X X X X X 0 0 0 0 0 0
0 0 X X X X X X 0 0 0 0 0 0 0 0
0 0 0 0 X X 0 0 0 0 0 0 0 0 0 0
```

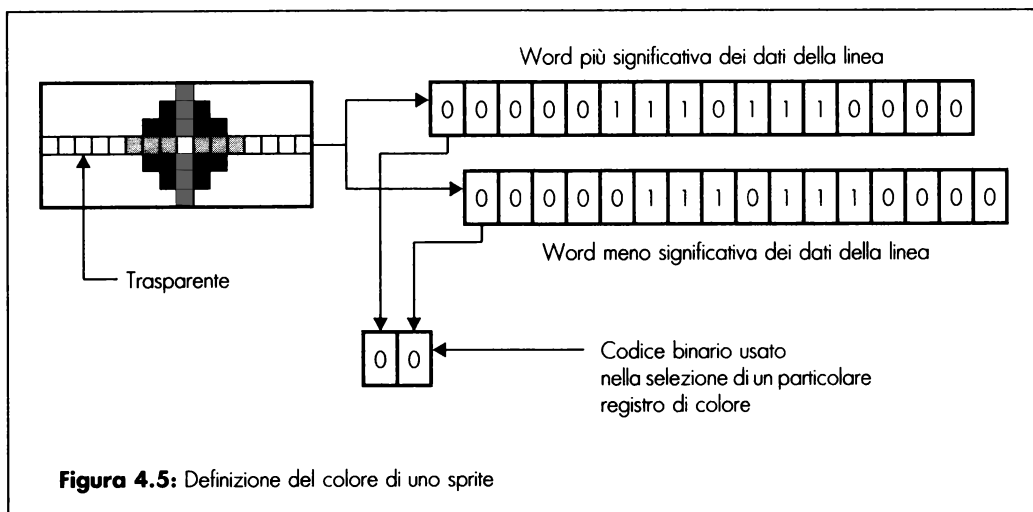
Figura 4.4: Sprite definito con l'immagine dell'astronave

In questo esempio, la larghezza massima dell'immagine è di dieci pixel e il suo orlo sinistro coincide con quello dello sprite. Se l'immagine da visualizzare è più stretta di 16 pixel, può essere collocata dove si preferisce all'interno dello sprite. Ad esempio, l'immagine considerata in Figura 4.4 potrebbe iniziare, invece che dal pixel 1, da uno qualunque nell'intervallo 2-7.

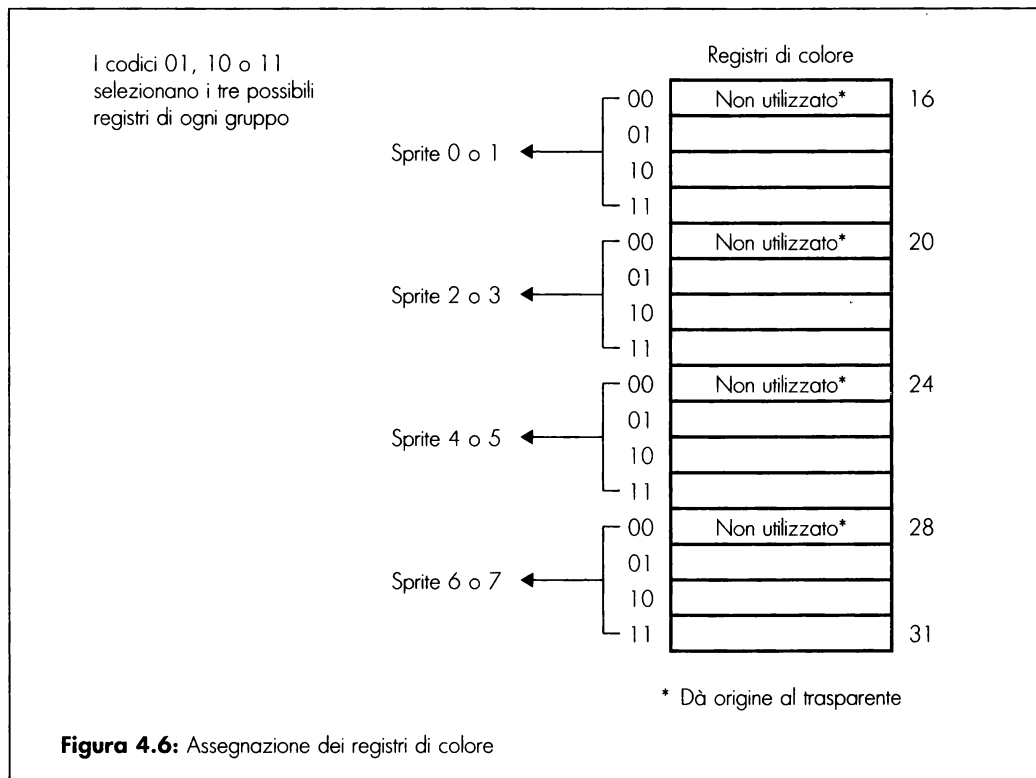
COLORE DI UNO SPRITE

Quando uno sprite viene usato singolarmente (cioè non è "collegato", come viene illustrato in seguito nel paragrafo "Sprite collegati"), i tre colori e il trasparente si possono usare per ogni pixel. I colori si scelgono più o meno come quelli di schermo. La Figura 4.5 mostra come viene determinato il colore di ogni pixel.

Gli 0 e gli 1 nelle due word di dati che definiscono le linee dello sprite, rappresentano nel

Figura 4.5: Definizione del colore di uno sprite

complesso un numero binario. A seconda del suo valore, viene selezionato uno dei quattro registri di colore assegnati dall'hardware a quel particolare sprite. Gli otto sprite utilizzano i registri di colore 16-31. Per quanto riguarda la selezione dei colori, gli sprite vengono organizzati in coppie, ognuna delle quali utilizza i registri mostrati nella Figura 4.6.

Il numero binario 00 viene usato in maniera particolare. Un pixel il cui valore sia 00 diventa trasparente, e lascia vedere lo sfondo o un altro sprite a priorità inferiore. Gli oggetti con priorità inferiore appaiono "dietro" gli oggetti con priorità superiore. Ogni sprite ha una priorità fissa rispetto agli altri. Può invece essere modificata la priorità fra gli sprite e i playfield. Si veda, al riguardo, il Capitolo 7.

Figura 4.6: Assegnazione dei registri di colore

DEFINIZIONE DI UNO SPRITE

Nella definizione di uno sprite può essere conveniente ricorrere a uno schizzo preliminare su carta. Indicando i colori con le cifre da 0 a 3, l'astronave dell'esempio precedente potrebbe apparire così:

```
0000122332210000
0001223333221000
0012223333222100
0001223333221000
0000122332210000
```

Il passo successivo consiste nel convertire le cifre 0-3 in numeri binari che verranno poi usati per costruire le word di dati vere e proprie.

COSTRUZIONE DELLA STRUTTURA DATI

Dopo aver stabilito la forma dello sprite, è necessario costruire la sua struttura dati, che sarà costituita da una serie di word in un'area continua di memoria. Alcune di queste word conterranno informazioni sulla posizione e il controllo dello sprite, altre informazioni sulla forma e sul colore. I passi da fare sono i seguenti:

- Scrivere la posizione orizzontale e verticale dello sprite nella prima word.
- Scrivere la posizione finale verticale nella seconda word.
- Tradurre i numeri 0-3 nei corrispondenti valori binari e usare questi valori per scrivere le word di dati da inserire nella struttura.
- Scrivere le word di controllo che indicano il termine della struttura dati.

I dati relativi agli sprite, come ogni altro dato che debba essere utilizzato dai chip custom, devono trovarsi nella memoria chip. Assicuratevi quindi che tutte queste strutture si trovino in una zona di memoria di tipo chip e allineate a un indirizzo pari.

La Tavola 4.1 mostra una struttura dati con l'indirizzo e la funzione di ogni word.

Tavola 4.1: Struttura dati di uno sprite

Indirizzo	Word	Funzione
N	Word di controllo 1	Posizione iniziale orizzontale e verticale
N+1	Word di controllo 2	Posizione finale verticale
N+2	Descrizione del colore - word meno significativa	Colore linea 1
N+3	Descrizione del colore - word più significativa	Colore linea 1
N+4	Descrizione del colore - word meno significativa	Colore linea 2
N+5	Descrizione del colore - word più significativa	Colore linea 2
	.	
	Word di fine dati	Due word che indicano il prossimo uso dello sprite

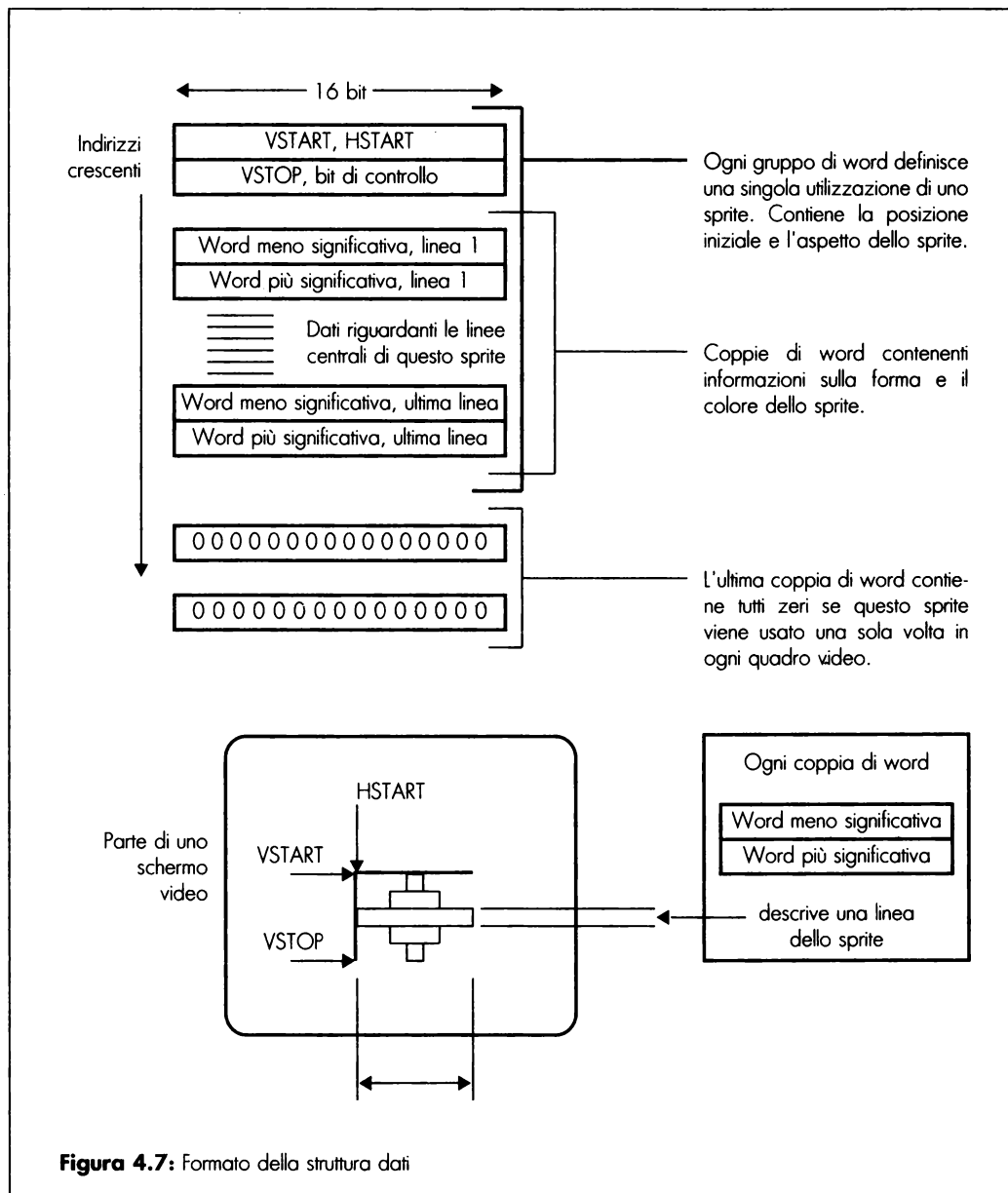
Tutti gli indirizzi relativi agli sprite sono indirizzi di word. È necessario un blocco continuo di memoria abbastanza ampio da contenere due word di controllo, due word per ogni linea orizzontale dello sprite e due word di fine dati.

Dal momento che questa struttura dati dev'essere accessibile da parte dei chip custom, deve trovarsi nella memoria chip.

Prima word di controllo: SPRxPOS

Questa word contiene la posizione iniziale verticale (VSTART) e orizzontale (HSTART) dello sprite. In pratica si tratta della posizione della prima linea dello sprite.

Bit 15-8 Gli 8 bit meno significativi di VSTART
Bit 7-0 Gli 8 bit più significativi di HSTART

Figura 4.7: Formato della struttura dati

Seconda word di controllo: SPRxCTL

Questa word contiene la posizione verticale finale dello sprite sullo schermo (VSTOP), cioè la linea *successiva* all'ultima linea dello sprite. Contiene inoltre altri dati relativi al collegamento degli sprite (di cui parleremo in seguito).

Bit 15-8 Gli 8 bit meno significativi di VSTOP
Bit 7 Usato nel collegamento degli sprite

Bit 6-3 Non utilizzati (azzerare)
Bit 2 Il bit più significativo di VSTART
Bit 1 Il bit più significativo di VSTOP
Bit 0 Il bit meno significativo di HSTART

Il valore VSTOP-VSTART indica (in linee) l'altezza dello sprite.

Word di dati del colore

Sono necessarie due word per descrivere ogni linea orizzontale dello sprite. Il totale delle word di dati sarà quindi pari al doppio dell'altezza dello sprite. La word più significativa contiene la cifra più a sinistra del numero binario. La word meno significativa contiene quella più a destra.

Per ottenere queste word di dati, occorre innanzitutto realizzare una rappresentazione dello sprite che indichi il colore di ogni pixel tramite una cifra da 0 a 3. Ogni numero rappresenta un registro di colore associato allo sprite. Ecco di nuovo la forma dell'astronave:

```
0000122332210000
0001223333221000
0012223333222100
0001223333221000
0000122332210000
```

Dopo si traduce ogni cifra in un numero binario. L'esempio che segue mostra la prima linea dello sprite. I numeri binari sono rappresentati verticalmente, con la cifra meno significativa nella linea superiore e quella più significativa in quella inferiore, per rispecchiare il modo in cui i dati si presentano in memoria.

```
0000100110010000 Word meno significativa
0000011111100000 Word piu' significativa
```

Ogni numero binario formato dalla combinazione delle due word individua un particolare registro di colore all'interno del gruppo assegnato a quel particolare sprite. Per esempio, lo sprite 0 utilizza i registri 17-19. I numeri binari corrispondenti a tali registri per il canale DMA dello sprite 0 sono elencati nella tavola che segue.

Tavola 4.2: Registri di colore di uno sprite

Numero binario	**Registro di colore**
00	Trasparente
01	17
10	18
11	19

Si ricordi che il valore binario 00 significa sempre "trasparente" e non si riferisce a un registro di colore specifico.

Word di fine dati

Quando la posizione verticale del pennello elettronico è uguale al valore di VSTOP nelle word di controllo di un particolare sprite, le due word successive della struttura dati vengono scritte nei registri di controllo dello sprite anziché nei registri di colore. Queste due word sono interpretate dall'hardware esattamente come quelle che erano state caricate in precedenza negli stessi registri. Se il valore di VSTART contenuto in queste word è inferiore a quello della posizione raggiunta dal pennello elettronico, lo sprite non verrà riutilizzato fino al quadro video successivo. Se non s'intende più riutilizzare uno sprite si deve indicare il valore 0. Più avanti torneremo sull'argomento della riutilizzazione degli sprite.

La struttura dati che segue rappresenta ancora una volta lo sprite dell'astronave. Lo sprite viene collocato nella posizione V = 65 e H = 128, che si trova nella parte visibile dello schermo.

```
SPRITE:
        DC.W    $6D60,$7200         ;VSTART, HSTART, VSTOP
        DC.W    $0990,$07E0         ;Prima coppia di word di dati
        DC.W    $13C8,$0FF0
        DC.W    $23C4,$1FF8
        DC.W    $13C8,$0FF0
        DC.W    $0990,$07E0
        DC.W    $0000,$0000         ;Fine dei dati
```

Visualizzazione di uno sprite

Dopo aver costruito la struttura-dati in memoria, il passo successivo consiste nell'informarne il sistema perché proceda alla visualizzazione. Questo paragrafo descrive l'uso degli sprite in modo "automatico", che consiste nell'utilizzare il canale DMA per la lettura e la visualizzazione dei dati (fino al raggiungimento della posizione VSTOP). L'uso "manuale" degli sprite viene trattato più avanti in questo stesso capitolo.

I passi da effettuare sono i seguenti:

1. Decidere quale canale di DMA degli sprite utilizzare (accertandosi che sia disponibile).

2. Impostare i puntatori ai dati.

3. Abilitare il DMA (se non è già abilitato).

4. Durante l'intervallo di vertical blanking, reimpostare i puntatori per i quadri video successivi.

Se il DMA relativo agli sprite viene disabilitato mentre vi è uno sprite visualizzato (cioè dopo VSTART ma prima di VSTOP), il sistema continuerà a visualizzare le ultime word di dati lette per tutte le linee di scansione successive. Ciò provoca la comparsa di una barra verticale sullo schermo. Si raccomanda perciò di disabilitare il DMA degli sprite soltanto durante l'intervallo di vertical blanking o in un momento nel quale si è *sicuri* che sullo schermo non c'è nessuno sprite.

SELEZIONE DI UN CANALE DMA E IMPOSTAZIONE DEI PUNTATORI

Nella scelta del canale DMA da utilizzare, è necessario tenere in considerazione i registri di colore assegnati a ogni sprite e la loro priorità relativa.

I canali di DMA degli sprite utilizzano due puntatori per la lettura dei dati e delle word di controllo. Durante l'intervallo di vertical blanking precedente al primo impiego di uno sprite, è necessario scrivere l'indirizzo della struttura dati corrispondente nei registri SPRxPTH e SPRxPTL ("x" indica il numero del canale DMA). SPRxPTH contiene la parte più significativa dell'indirizzo e SPRxPTL i rimanenti 16 bit. Il bit meno significativo di SPRxPTL viene ignorato, dal momento che i dati devono comunque iniziare a un indirizzo pari. Come di consueto, questi registri possono essere considerati dal 68000 come un'unica long word.

Nell'esempio che segue, il 68000 inizializza all'indirizzo $20000 i puntatori allo sprite 0. In genere questo è un compito svolto dal Copper.

```
MOVE.L  #$20000,SPR0PTH+CUSTOM   ;SPR0PT = $20000
```

Questi puntatori sono dinamici: vengono incrementati durante la lettura in modo che puntino sempre alla word successiva. Dopo aver letto le word di controllo e averle collocate negli appropriati registri, vengono lette le word che descrivono la forma e i colori dello sprite. Queste word vengono trasferite nei registri dati degli sprite, e vengono quindi utilizzate dai canali DMA per produrre l'immagine sullo schermo. Si veda anche il paragrafo "Dettagli sull'hardware degli sprite".

REIMPOSTAZIONE DEI PUNTATORI

Per ogni quadro video, il sistema legge automaticamente i dati relativi allo sprite e ne genera l'immagine sullo schermo. Se si desidera che l'immagine venga visualizzata anche nei quadri video successivi, è necessario reimpostare i puntatori allo sprite durante l'intervallo di vertical blanking. Con ogni quadro video, infatti, i puntatori vengono via via incrementati per puntare ai successivi dati di visualizzazione.

Questa riscrittura diventa uno dei compiti della routine di vertical blanking, e può essere comodamente gestita tramite la lista di istruzioni del Copper.

ESEMPIO DI VISUALIZZAZIONE DI UNO SPRITE

Riportiamo come esempio il codice necessario per visualizzare nella posizione V = 65, H = 128 l'immagine dell'astronave. Anche in questo caso è necessario includere il file "hw_examples.i" contenuto nell'Appendice J.

```
;
;Preparazione di un bitplane
;
      LEA     CUSTOM,a0            ;Base dei chip custom
      MOVE.W  #$1200,BPLCON0(a0)   ;1 bitplane
      MOVE.W  #$0000,BPL1MOD(a0)   ;Modulo = 0
      MOVE.W  #$0000,BPLCON1(a0)   ;Scroll orizzontale = 0
      MOVE.W  #$0024,BPLCON2(a0)   ;Priorita' degli sprite
      MOVE.W  #$0038,DDFSTRT(a0)
      MOVE.W  #$00D0,DDFSTOP(a0)
```

```
      MOVE.W  #$2C81,DIWSTRT(a0)   ;Inizio finestra video
      MOVE.W  #$2CC1,DIWSTOP(a0)   ;Fine finestra video

      MOVE.W  #$0008,COLOR00(a0)   ;Sfondo = blu scuro
      MOVE.W  #$0000,COLOR01(a0)   ;Primo piano = nero
      MOVE.W  #$0FF0,COLOR17(a0)   ;Colore 17 = giallo
      MOVE.W  #$00FF,COLOR18(a0)   ;Colore 18 = cyan
      MOVE.W  #$0F0F,COLOR19(a0)   ;Colore 19 = magenta
;
;Istruzioni del Copper a $20000
;
      MOVE.L  #$20000,a1           ;Destinazione
      LEA     COPPERL(pc),a2       ;Dati
CLOOP:
      MOVE.L  (a2),(a1)+          ;Trasferisce una long word
      CMPI.L  #$FFFFFFFE,(a2)+    ;Fine della lista?
      BNE.S   CLOOP
;
;Trasferisce lo sprite a $27000
;
      MOVE.L  #$27000,a1           ;Destinazione
      LEA     SPRITE(pc),a2        ;Dati
SPRLOOP:
      MOVE.L  (a2),(a1)+          ;Trasferisce una long word
      CMPI.L  #$00000000,(a2)+    ;Fine dei dati?
      BNE.S   SPRLOOP
;
;Collocazione di uno sprite fasullo a $30000
;Dal momento che tutti gli sprite vengono attivati
;contemporaneamente e noi ne utilizziamo uno solo, i rimanenti
;vengono puntati a questi dati fasulli
;
      MOVE.L  #$00000000,$30000
;
;Inizializza il Copper
;
      MOVE.L  #$20000,COP1LC(a0)
;
;Riempie il bitplane di $FFFFFFFF
;
      MOVE.L  #$21000,a1           ;Indirizzo del bitplane
      MOVE.W  #2559,d0             ;2560 long word = 10240 byte
FLOOP:
      MOVE.L  #$FFFFFFFF,(a1)+    ;Riempie una long word
      DBRA    d0,FLOOP
;
;Abilita il DMA
;
      MOVE.W  d0,COPJMP1(a0)       ;Reset del Copper
      MOVE.W  #$83A0,DMACON(a0)    ;Bitplane, Copper e sprite
      RTS                          ;...resto del programma
```

```
COPPERL:
        DC.W    BPL1PTH,$0002       ;Bitplane 1 = $21000
        DC.W    BPL1PTL,$1000
        DC.W    SPR0PTH,$0002       ;Sprite 0 = $27000
        DC.W    SPR0PTL,$7000
        DC.W    SPR1PTH,$0003       ;Sprite 1 = $30000
        DC.W    SPR1PTL,$0000
        DC.W    SPR2PTH,$0003       ;Sprite 2 = $30000
        DC.W    SPR2PTL,$0000
        DC.W    SPR3PTH,$0003       ;Sprite 3 = $30000
        DC.W    SPR3PTL,$0000
        DC.W    SPR4PTH,$0003       ;Sprite 4 = $30000
        DC.W    SPR4PTL,$0000
        DC.W    SPR5PTH,$0003       ;Sprite 5 = $30000
        DC.W    SPR5PTL,$0000
        DC.W    SPR6PTH,$0003       ;Sprite 6 = $30000
        DC.W    SPR6PTL,$0000
        DC.W    SPR7PTH,$0003       ;Sprite 7 = $30000
        DC.W    SPR7PTL,$0000
        DC.W    $FFFF,$FFFE         ;Fine della lista
;
;Dati dello sprite
;
SPRITE:
        DC.W    $6D60,$7200         ;VSTART, HSTART, VSTOP
        DC.W    $0990,$07E0         ;Prima coppia di word di dati
        DC.W    $13C8,$0FF0
        DC.W    $23C4,$1FF8
        DC.W    $13C8,$0FF0
        DC.W    $0990,$07E0
        DC.W    $0000,$0000         ;Fine dei dati
```

Movimento di uno sprite

Uno sprite generato in modo automatico può essere spostato indicando nella sua struttura dati una diversa posizione. Per ogni quadro video, i dati vengono riletti e lo sprite ridisegnato. Perciò, se se ne cambia la posizione prima che venga ridisegnato, lo sprite sembra spostarsi sullo schermo.

Nel compiere quest'operazione, bisogna avere cura di non modificare la posizione dello sprite proprio mentre i canali DMA la stanno leggendo per sapere dove visualizzarlo, per evitare che venga letto il valore di VSTART di un quadro video e il valore di VSTOP del successivo. Ciò potrebbe originare una certa confusione sullo schermo. Un momento sicuro per effettuare questo cambiamento è l'intervallo di vertical blanking. Il movimento degli sprite potrebbe perciò diventare uno dei compiti del Copper, come si vede nell'esempio seguente.

Muovendosi attraverso lo schermo, gli sprite possono entrare in collisione con altri sprite o con i playfield. È possibile utilizzare l'hardware dell'Amiga per rilevare queste collisioni e farne uso per realizzare effetti speciali o semplicemente per mantenere gli oggetti in movimento all'interno di limiti precisi. Questo argomento è trattato più ampiamente nel Capitolo 7.

Nell'esempio che segue, lo sprite dell'astronave viene fatto rimbalzare da una parte all'altra

dello schermo, in modo che cambi direzione ogni volta che urta contro un bordo.

La posizione dello sprite, assegnata da VSTART e VSTOP, si trova all'indirizzo $27000. VSTOP si trova all'indirizzo $27002. Per far muovere lo sprite si deve scrivere in queste locazioni. Una volta per ogni quadro video, VSTART viene incrementato (o decrementato) di 1 e HSTART di 2. Quindi viene calcolato il nuovo valore di VSTOP = VSTART + 6.

```
      MOVE.B  #151,d0               ;Contatore orizzontale
      MOVE.B  #194,d1               ;Contatore verticale
      MOVE.B  #64,d2                ;Posizione orizzontale
      MOVE.B  #44,d3                ;Posizione verticale
      MOVE.B  #1,d4                 ;Incremento orizzontale
      MOVE.B  #1,d5                 ;Incremento verticale
;
;A questo punto entriamo in attesa dell'inizio del nuovo
;quadro video, per avere un'immagine regolare.
;
      LEA     CUSTOM,a0             ;Base dei chip custom
VLOOP:
      MOVE.B  VHPOSR(a0),d6         ;Posizione verticale
      CMPI.B  #$20,d6               ;Termine dello schermo?
      BNE.S   VLOOP
;
;In alternativa si puo' usare il codice seguente:
;
;     LEA     CUSTOM,a0             ;Base dei chip custom
;LOOP:
;     MOVE.W  INTREQR(a0),d6        ;Legge le richieste di interrupt
;     AND.W   #$0020,d6             ;Maschera tutto eccetto il bit di VBLANK
;     BEQ.S   VLOOP                 ;Fino a che il bit non e' a 1
;     MOVE.W  #$0020,INTREQ(a0)     ;Reimposta il bit
;
;Questo metodo funziona solo se l'interrupt relativo
;all'intervallo di vertical blanking e' stato disabilitato,
;il che non e' consigliabile eccetto che per periodi molto brevi.
;
      ADD.B   d4,d2                 ;Incrementa la posizione orizzontale
      SUBQ.B  #1,d0                 ;Decrementa il contatore orizzontale
      BNE.S   L1
      MOVE.B  #151,d0               ;Ripristina il contatore
      EORI.B  #$FE,d4               ;Inverte la direzione
L1:   MOVE.B  d2,$27001             ;Scrive HSTART
      ADD.B   d5,d3                 ;Incrementa la posizione verticale
      SUBQ.B  #1,d1                 ;Decrementa il contatore verticale
      BNE.S   L2
      MOVE.B  #194,d1               ;Ripristina il contatore
      EORI.B  #$FE,d5               ;Inverte la direzione
L2:   MOVE.B  d3,$27000             ;Scrive VSTART
      MOVE.B  d3,d6
      ADD.B   #6,d6                 ;VSTOP = VSTART + 6
      MOVE.B  d6,$27002             ;Scrive VSTOP
      BRA.S   VLOOP                 ;Per sempre...
```

Creazione di sprite addizionali

Per utilizzare più di uno sprite, è necessario creare una struttura dati per ognuno e controllarne la visualizzazione tramite i puntatori appropriati: SPR1PTH e SPR1PTL per lo sprite 1, SPR2PTH e SPR2PTL per lo sprite 2 e così via.

Abilitando il DMA relativo agli sprite, vengono abilitati tutti gli sprite, che entrano così in modo automatico. Non è necessario ripetere l'abilitazione più di una volta.

Una volta che viene abilitato il DMA degli sprite, tutti e otto i puntatori agli sprite *devono* essere inizializzati per puntare a uno sprite reale o a dati che definiscano uno sprite nullo. Uno sprite non inizializzato correttamente può causare la comparsa di dati spurî sullo schermo.

Si rammenti che alcuni sprite diventano inservibili quando vengono utilizzati più cicli DMA del consueto, per esempio definendo una finestra video più ampia del normale o richiedendo lo scroll orizzontale di un playfield. Si veda anche la Figura 6.9.

Si rammenti inoltre che ogni coppia di sprite utilizza colori contenuti in diverse serie di registri di colore, come mostra la Tavola 4.3.

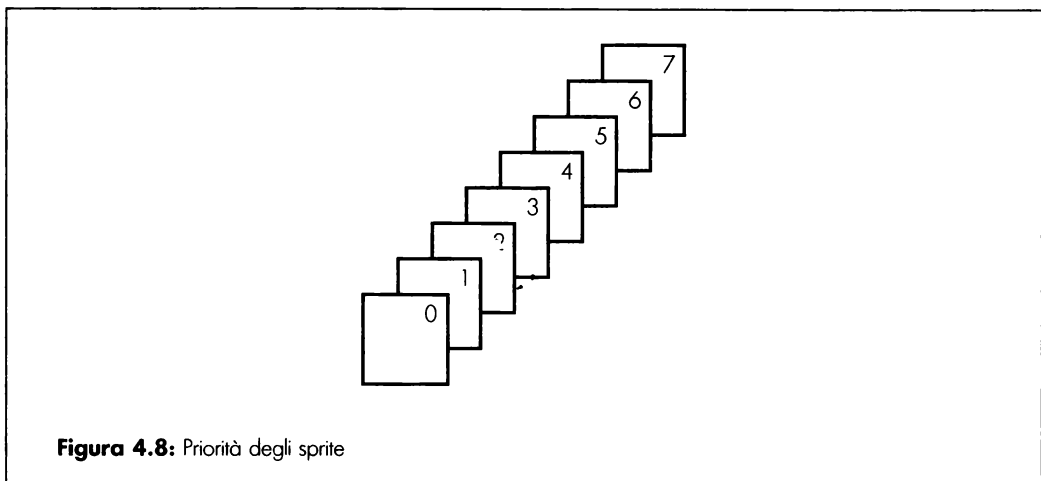
Tavola 4.3: Registri di colore per coppie di sprite

Sprite	Registri di colore
0 e 1	17-19
2 e 3	21-23
4 e 5	25-27
6 e 7	29-31

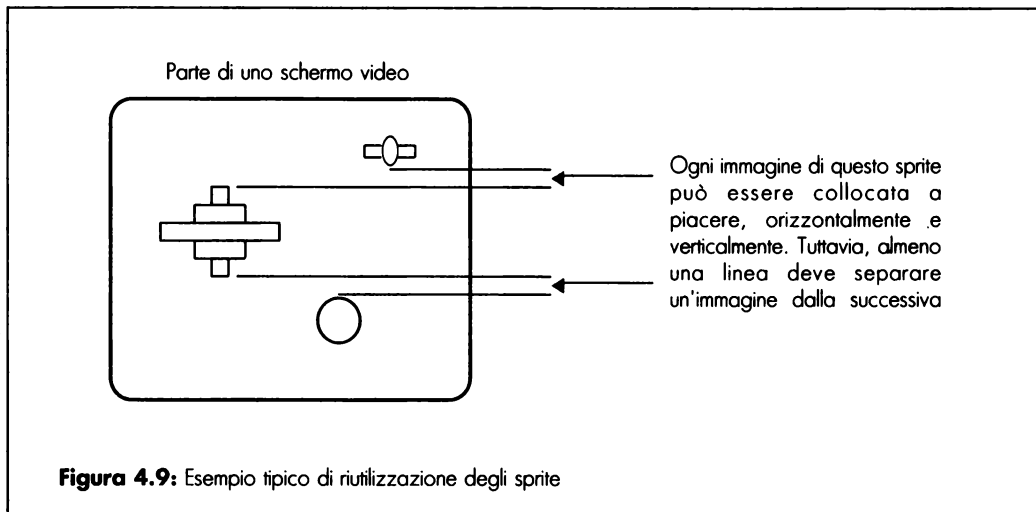
PRIORITÀ DEGLI SPRITE

Quando vi sono diversi sprite sullo schermo, si deve cominciare a prendere in considerazione la loro priorità relativa, ovvero l'ordine in cui devono apparire l'uno davanti all'altro. Ogni sprite ha una priorità relativa fissa; lo sprite avente il numero d'ordine più basso ha la priorità più alta, e di conseguenza appare di fronte a tutti gli altri.

Si veda anche il Capitolo 7.

Figura 4.8: Priorità degli sprite

Riutilizzazione dei canali DMA di uno sprite

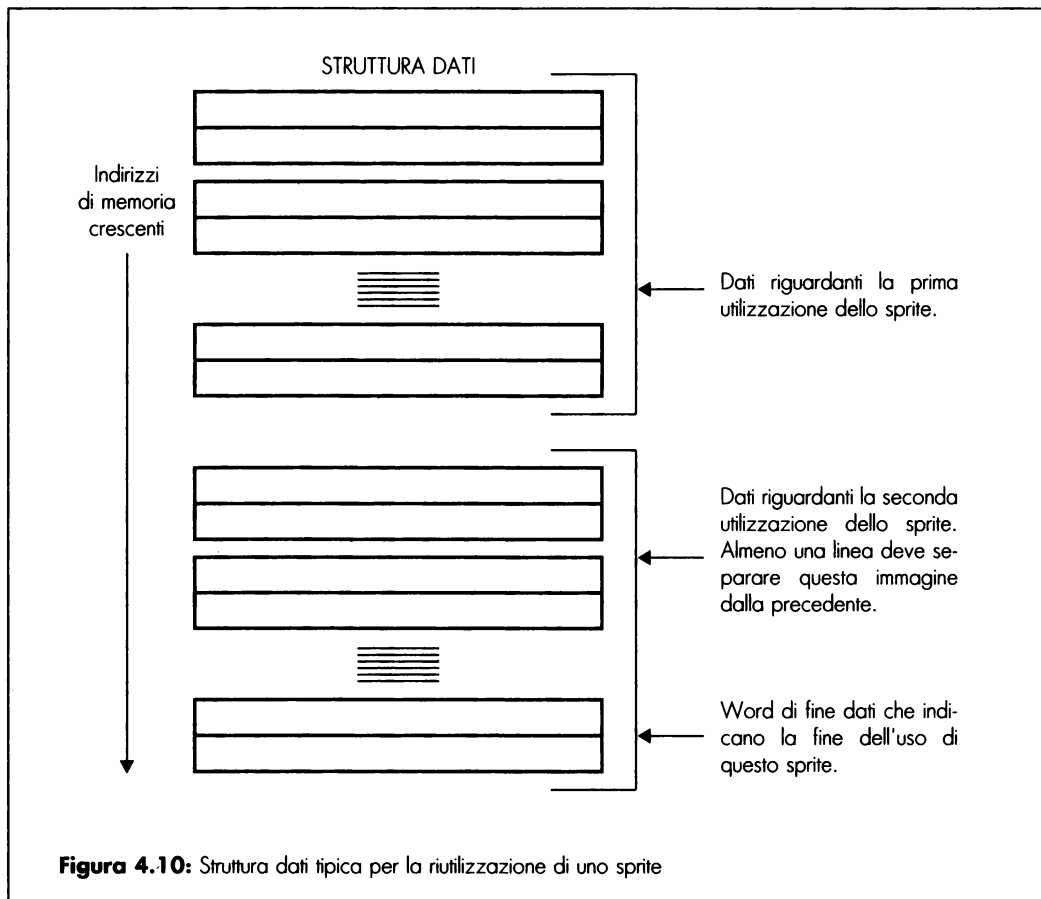
Ogni canale DMA relativo agli sprite è in grado di produrre più di un'immagine per ogni quadro video. Può infatti essere necessario, a volte, avere più di otto oggetti sullo schermo oppure può accadere che il numero degli sprite disponibili sia stato ridotto dal fatto di averne "collegati" alcuni per ottenere oggetti con più colori o dal fatto di averli affiancati o sovrapposti per ottenere oggetti più complessi. Come mostra la Figura 4.9 ogni canale DMA può essere riutilizzato parecchie volte per ogni quadro video.

Figura 4.9: Esempio tipico di riutilizzazione degli sprite

Quando si utilizza un solo sprite, al termine della sua struttura dati si collocano due word formate soltanto da zeri, per indicare al canale DMA di cessare il trasferimento dei dati. Per riutilizzare ancora lo stesso sprite, è sufficiente sostituire queste due word con una completa struttura dati che ne descriva il comportamento in una posizione successiva dello schermo. Le due word contenenti il valore zero vanno utilizzate soltanto al termine dell'ultima struttura dati. La figura 4.10 mostra la struttura dati dell'immagine precedente.

La sola restrizione da tener presente, per riutilizzare più volte uno sprite nello stesso quadro video, è che la linea inferiore di ogni esemplare dello sprite dev'essere separata almeno di una linea di scansione dall'immagine seguente. Ciò è dovuto al fatto che ogni canale DMA degli sprite può leggere soltanto due word per riga. La linea intermedia serve al canale DMA per trasferire nei registri appropriati le due word di controllo.

L'esempio seguente mostra l'immagine dell'astronave, poi lo ridisegna con alcune modifiche. Dal momento che le differenze riguardano soltanto lo sprite, qui riportiamo solo i dati. Tuttavia, l'immagine appare migliore se il contenuto dei registri di colore è quello proposto.

```
        LEA     CUSTOM,a0
        MOVE.W  #$0F00,COLOR17(a0)    ;Rosso
        MOVE.W  #$0FF0,COLOR18(a0)    ;Giallo
        MOVE.W  #$0FFF,COLOR19(a0)    ;Bianco
SPRITE:
        DC.W    $6D60,$7200
        DC.W    $0990,$07E0
```

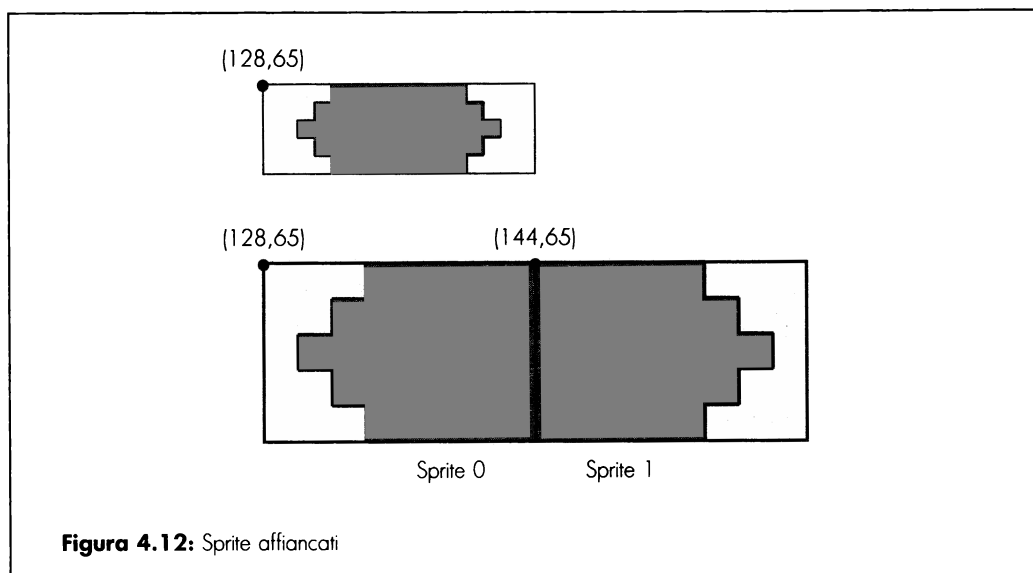
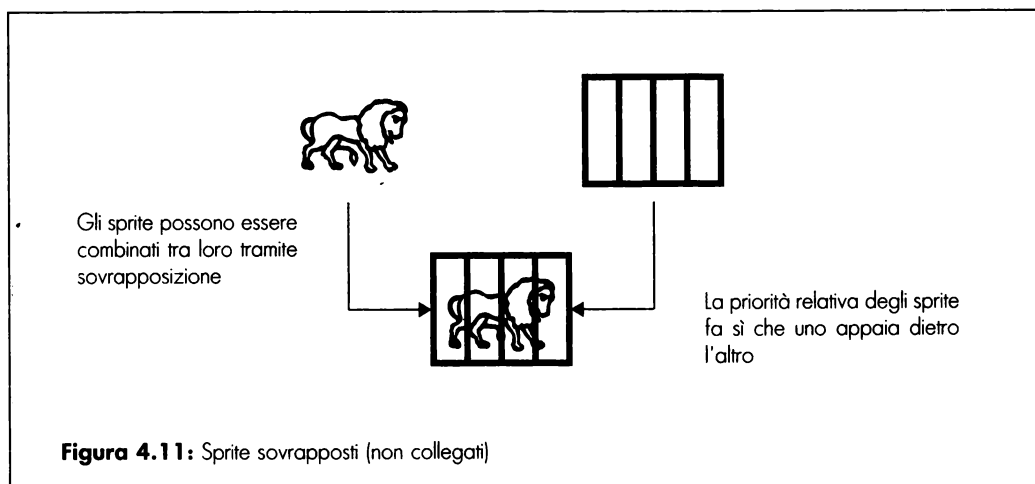
Figura 4.10: Struttura dati tipica per la riutilizzazione di uno sprite

```
DC.W    $13C8,$0FF0
DC.W    $23C4,$1FF8
DC.W    $13C8,$0FF0
DC.W    $0990,$07E0
DC.W    $8080,$8D00          ;VSTART, HSTART e VSTOP
DC.W    $1818,$0000
DC.W    $7E7E,$0000
DC.W    $7FFE,$0000
DC.W    $FFFF,$2000
DC.W    $FFFF,$2000
DC.W    $FFFF,$3000
DC.W    $FFFF,$3000
DC.W    $7FFE,$1800
DC.W    $7FFE,$0C00
DC.W    $3FFC,$0000
DC.W    $0FF0,$0000
DC.W    $03C0,$0000
DC.W    $0180,$0000
DC.W    $0000,$0000          ;Fine dei dati
```

Sprite sovrapposti o affiancati

Per ottenere immagini più complesse o più larghe dei normali 16 pixel, è possibile sovrapporre diversi sprite. Ciò significa semplicemente che le loro posizioni sullo schermo coincidono o sono molto vicine. Posizioni ravvicinate possono dare l'impressione che l'oggetto sia più largo di 16 pixel.

La priorità tra gli sprite fa in modo che uno dei due appaia sempre in secondo piano, quando sono sovrapposti. I relativi circuiti danno allo sprite con il numero più basso la priorità più alta. Perciò, nel disegnare immagini costituite da sprite sovrapposti, assicuratevi che lo sprite in primo piano abbia un numero più basso di quello in secondo piano. Nella Figura 4.11, per esempio, la gabbia dev'essere originata da uno sprite di numero inferiore rispetto a quello del leone.

Figura 4.11: Sprite sovrapposti (non collegati)

Figura 4.12: Sprite affiancati

È possibile creare sprite più larghi di 16 pixel collocandone vicini due (o più). Per esempio, l'astronave della Figura 4.12 può diventare grande il doppio se si utilizzano due sprite affiancati.

Sprite collegati

Tramite il "collegamento" di due sprite si riesce a crearne uno che consente di scegliere tra 15 colori (più il trasparente). Ecco le operazioni da svolgere per raggiungere questo obiettivo.

- Si devono utilizzare due canali di DMA, creando due sprite delle stesse dimensioni e collocati nella stessa posizione.
- Si deve impostare a 1 il bit ATTACH nella word di controllo del secondo sprite.

I 15 colori disponibili corrispondono all'intera gamma dei registri disponibili per gli sprite, vale a dire i registri 17-31. La maggior varietà diventa possibile perché ogni pixel contiene quattro bit d'informazione, anziché i normali due. Ogni sprite della coppia contribuisce con due bit al numero binario che rappresenta il colore. Se per esempio si utilizzano i canali 0 e 1, le due word che costituiscono i dati della linea 1 nel primo sprite si uniscono alle due word che svolgono la stessa funzione nel secondo sprite, e si combinano in quattro word che costituiscono i dati della linea 1 dell'oggetto risultante.

Gli sprite possono essere collegati nelle seguenti combinazioni:

Lo sprite 1 allo sprite 0
Lo sprite 3 allo sprite 2
Lo sprite 5 allo sprite 4
Lo sprite 7 allo sprite 6

Tutti questi collegamenti possono essere attivi durante il medesimo quadro video, anche contemporaneamente. Per esempio, supponiamo di aver bisogno di un numero maggiore di colori nell'immagine dell'astronave vista in precedenza, e di poter usare a questo scopo gli sprite 0 e 1. In questo sprite ci sono cinque colori più il trasparente.

```
0000154444510000
0001564444651000
0015676446765100
0001564444651000
0000154444510000
```

La prima linea richiede le quattro word di dati mostrate nella Tavola 4.4.

Tavola 4.4: Dati della prima linea dello sprite dell'astronave

	Pixel															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Linea 1	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
Linea 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Linea 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Linea 4	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0

Lo sprite avente il numero più alto (in questo caso lo sprite 1) fornisce le word più significative. Perciò, le word di cui sopra andranno inserite nelle strutture dati nel modo seguente:

Linea 1 Word più significativa della prima linea dello sprite 1
Linea 2 Word meno significativa della prima linea dello sprite 1
Linea 3 Word più significativa della prima linea dello sprite 0
Linea 4 Word meno significativa della prima linea dello sprite 0

A proposito dell'ordine in cui queste word sono collocate in memoria, si veda anche la Figura 4.7. Si tenga presente che si tratta di due distinte strutture dati.

Tavola 4.5: Registri di colore per gli sprite collegati

Numero decimale	Numero binario	Registro di colore
0	0000	trasparente
1	0001	17
2	0010	18
3	0011	19
4	0100	20
5	0101	21
6	0110	22
7	0111	23
8	1000	24
9	1001	25
10	1010	26
11	1011	27
12	1100	28
13	1101	29
14	1110	30
15	1111	31

Il collegamento è effettivamente operante solo quando il bit 7 (ATTACH) della seconda word di controllo è impostato a 1 nella struttura appartenente allo sprite con numero dispari. In questo esempio si tratta del bit 7 della seconda word di controllo dello sprite 1.

Se gli sprite vengono spostati, deve comunque essere mantenuta invariata la loro posizione

relativa. In caso contrario i pixel cambiano colore. Entrambi gli sprite tornano a mostrare 3 colori più il trasparente, ma i colori sono diversi da quelli che avrebbero se non fossero stati collegati: lo sprite col numero più basso utilizza i registri di colore 17-19, mentre quello col numero più alto utilizza i registri 20, 24 e 28.

Le seguenti strutture dati descrivono l'astronave a sei colori formata dai due sprite collegati.

```
SPRITE0:
      DC.W    $6D60,$7200         ;VSTART = 65, HSTART = 128
      DC.W    $0C30,$0000
      DC.W    $1818,$0420
      DC.W    $342C,$0E70
      DC.W    $1818,$0420
      DC.W    $0C30,$0000
      DC.W    $0000,$0000         ;Fine dello sprite 0
SPRITE1:
      DC.W    $6D60,$7280         ;Come lo sprite 0, ma con il bit ATTACH impostato
      DC.W    $07E0,$0000
      DC.W    $0FF0,$0000
      DC.W    $1FF8,$0000
      DC.W    $0FF0,$0000
      DC.W    $07E0,$0000
      DC.W    $0000,$0000         ;Fine dello sprite 1
```

Uso manuale degli sprite

È sempre consigliabile utilizzare gli sprite tramite i canali DMA automatici. Qualche volta, tuttavia, può essere utile impostare questi registri direttamente con il microprocessore. Gli sprite possono essere attivati "manualmente" ogniqualvolta non sono usati dai canali DMA. Uno sprite che in alto sullo schermo mostra un'immagine tramite DMA, per esempio, può essere riutilizzato manualmente per visualizzare una barra verticale colorata più in basso. Gli sprite possono essere impiegati in questo modo anche quando il DMA è disabilitato.

L'attivazione avviene impostando i registri SPRxDATB e SPRxDATA, in quest'ordine. SPRxDATA va scritto per ultimo perché l'accesso a questo registro dà il via alla generazione dell'immagine durante il successivo confronto orizzontale. I dati scritti saranno quindi visualizzati a ogni linea successiva alla posizione specificata nei registri SPRxPOS e SPRxCTL. Se i dati rimangono gli stessi questo dà origine a una barra verticale, ma se vengono riscritti per ogni linea può essere generata un'immagine più complessa.

Uno sprite può essere disattivato scrivendo nel registro SPRxCTL. Scrivendo invece nel registro SPRxPOS, lo sprite può essere mosso manualmente in qualunque momento, anche durante il normale uso degli sprite.

Dettagli sull'hardware degli sprite

Gli sprite sono generati dalla circuiteria mostrata in Figura 4.13. Tramite un diagramma a blocchi questa figura mostra in che modo un paio di word di dati diventano una serie di pixel sullo schermo.

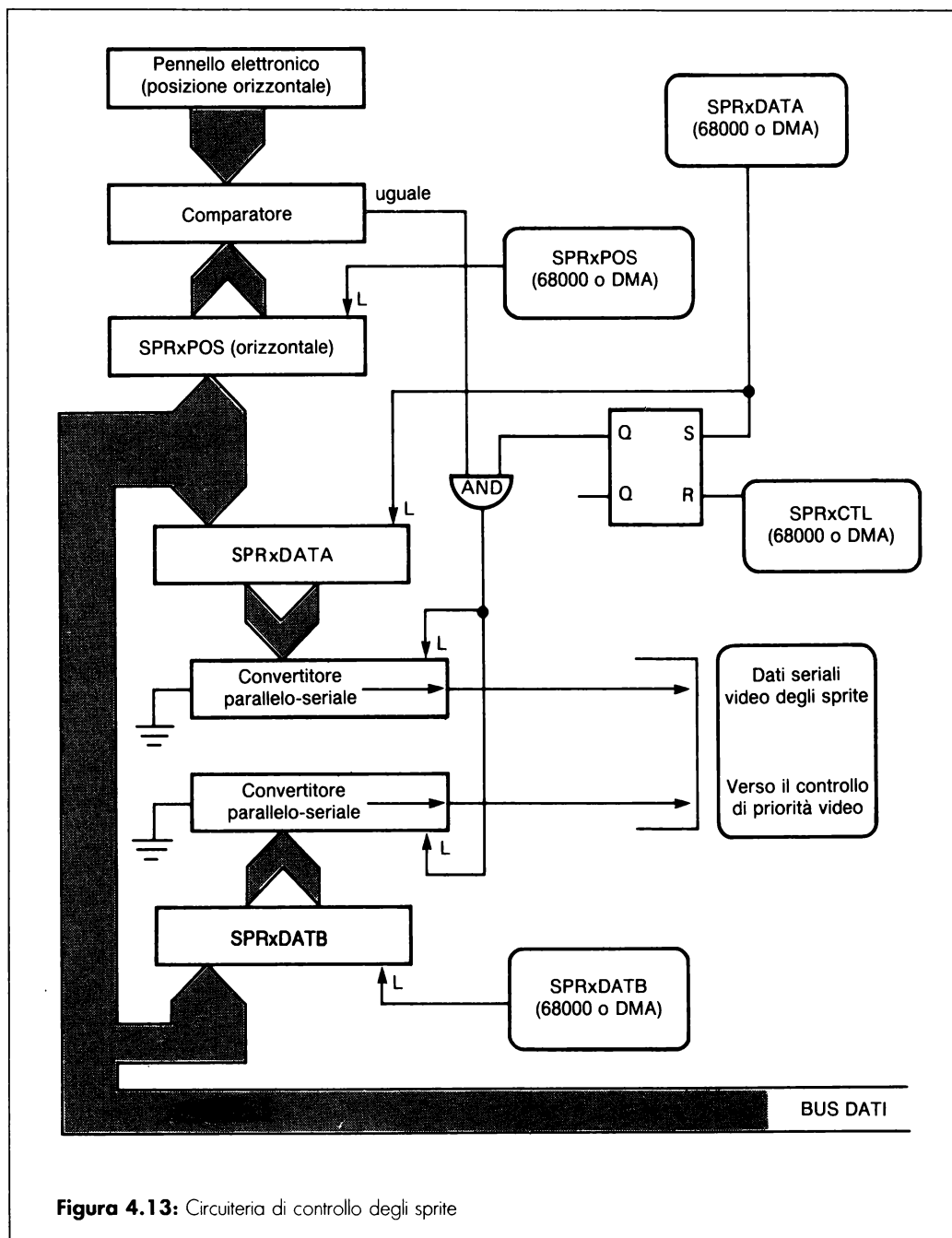



Figura 4.13: Circuiteria di controllo degli sprite

- Registri dati. I registri SPRxDATA e SPRxDATB contengono i bit che descrivono una linea orizzontale per ognuno degli otto sprite. Ogni linea è larga 16 pixel, e ogni linea è definita da due word per consentire la scelta dei tre colori più il trasparente.

- Convertitori parallelo-seriali. Ognuno dei 16 bit di dati viene inviato individualmente alla circuiteria di selezione del colore nel momento in cui il pixel a esso associato dev'essere visualizzato sullo schermo. Appena i dati vengono trasmessi dai registri, i convertitori iniziano a farli "scorrere" in modo che vengano trasferiti prima i bit più a sinistra. Un singolo trasferimento avviene in un intervallo di tempo equivalente alla visualizzazione di un pixel in bassa risoluzione, e continua fino a che tutti e 16 i bit non sono stati inviati alla circuiteria video. Il nuovo trasferimento non ha inizio finché i convertitori non sono nuovamente caricati tramite i registri dati. Dal momento che l'immagine video viene prodotta da un pennello elettronico che si muove sullo schermo da sinistra a destra, la struttura dei dati corrisponde esattamente all'immagine, vale a dire con i dati più significativi a sinistra.

- Dati seriali video degli sprite. Questi dati vengono sottoposti ad appositi circuiti di controllo per stabilire la priorità tra sprite e playfield.

- Registri di posizione. Questi registri, chiamati SPRxPOS, contengono il valore della posizione orizzontale (X) e verticale (Y) degli otto sprite.

- Registri di controllo. Questi registri, chiamati SPRxCTL, contengono la posizione di arresto e le informazioni di collegamento per gli otto sprite.

- Registro di posizione del pennello elettronico. Questo registro tiene informato il sistema sulla posizione del pennello elettronico.

- Comparatore. Questo circuito confronta il valore della posizione del pennello elettronico con la posizione iniziale dello sprite contenuta nel registro SPRxPOS. Quando il pennello raggiunge la posizione corrispondente al pixel superiore sinistro dello sprite, il comparatore invia un segnale ai convertitori e inizia la visualizzazione dello sprite.

La Figura 4.13 permette inoltre di notare che:

- L'accesso in scrittura al registro di controllo disabilita il comparatore. Questo impedisce il passaggio dei dati dai registri dati ai convertitori parallelo-seriali e quindi da questi ultimi allo schermo.

- L'accesso in scrittura al registro dati A abilita il comparatore. Questo consente la visualizzazione dei dati quando la posizione orizzontale del pennello elettronico è uguale a quella contenuta nel registro SPRxPOS.

- Se il comparatore è abilitato, i dati vengono inviati allo schermo, con il pixel più a sinistra in corrispondenza della posizione orizzontale definita in SPRxPOS.

- Finché il comparatore rimane abilitato, il contenuto dei registri dati viene visualizzato nella corretta posizione orizzontale in ogni linea.

- I dati contenuti nei registri dati non cambiano. Devono essere riscritti dal 68000 oppure tramite il DMA.

Gli elementi ora descritti servono per la creazione automatica degli sprite, che avviene nel modo seguente. Quando gli sprite sono controllati tramite DMA, il puntatore a 18 bit contenuto nei registri SPRxPTH e SPRxPTL viene usato per leggere le prime due word della struttura dati. Queste word, che contengono la posizione iniziale e quella finale dello sprite, vengono trasferite nei registri SPRxPOS e SPRxCTL. A questo punto, i registri SPRxPTH e SPRxPTL puntano alla prima word di dati dello sprite (word meno significativa della linea 1 dello sprite).
L'aver scritto nel registro SPRxCTL ha quindi disabilitato lo sprite. Il canale DMA relativo allo sprite entra ora in attesa finché il valore della posizione verticale del pennello elettronico non è uguale al valore di VSTART (Y). A questo punto viene abilitato l'accesso ai dati.
Il canale DMA esamina il valore di VSTOP (contenuto in SPRxCTL), che individua la linea successiva all'ultima dello sprite, e il valore di VSTART (contenuto in SPRxPOS), per sapere quante linee di dati saranno trasferite. Per ogni linea vengono utilizzate due word. La prima viene memorizzata in SPRxDATA e la seconda in SPRxDATB.
Il trasferimento avviene, per ogni linea, durante l'intervallo di horizontal blanking, prima dell'inizio della finestra video. Ciò abilita il circuito comparatore che dà il via al trasferimento dei dati non appena viene raggiunta la posizione orizzontale specificata da HSTART (contenuto in SPRxPOS).
Se VSTOP-VSTART è uguale a zero, non si verifica alcun output di dati. La successiva coppia di word viene comunque letta, ma non viene trasferita nei registri dati. Fornisce invece i nuovi valori di SPRxPOS e SPRxCTL.
Quando uno sprite viene usato una sola volta nello stesso quadro video, la coppia di word finale viene comunque caricata nei registri SPRxPOS e SPRxCTL. Perciò se si tratta dell'unico (o dell'ultimo) uso di quello sprite, entrambe le word devono essere azzerate.

Sommario dei registri relativi agli sprite

Vi sono otto serie complete di registri utilizzate dagli sprite. Ogni serie è formata da sei registri. Descriviamo soltanto i registri riguardanti lo sprite 0, dal momento che gli altri – a parte il nome – sono del tutto identici.

PUNTATORI

Sono i registri usati dal sistema per individuare i dati da utilizzare. Durante la visualizzazione di uno sprite, questi registri devono puntare sempre alla successiva word da trasferire, perciò vanno reimpostati all'inizio di ogni intervallo di vertical blanking.

SPR0PTH e SPR0PTL

Questa coppia di registri contiene l'indirizzo dei dati relativi allo sprite 0.

REGISTRI DI CONTROLLO

SPR0POS

Questo è il registro che individua la posizione dello sprite 0. La word in esso contenuta identifica la posizione in cui andrà collocato l'angolo superiore sinistro dello sprite (ovvero il bit più significativo della prima word di dati).

La risoluzione della posizione degli sprite è quella di uno schermo intero: 320 x 256 (320 x 200 in NTSC), ed è indipendente da quella del playfield sottostante.

- I bit 15-8 specificano la coordinata verticale iniziale: V7-V0.
- I bit 7-0 specificano la coordinata orizzontale iniziale: H8-H1.

SPR0CTL

Questo registro in genere viene usato solo dal canale DMA dello sprite. Contiene informazioni di controllo riguardanti il trasferimento dei dati.

- I bit 15-8 specificano la coordinata verticale finale dello sprite: V7-V0.
- Il bit 7 è il bit di collegamento, e ha significato soltanto per gli sprite dispari. Indica che gli sprite 0 e 1 (oppure 2 e 3, 4 e 5, o 6 e 7) devono essere considerati collegati per quanto riguarda la selezione dei colori, assumendo quindi una profondità di 4 bit. Lo sprite dispari (quello con il numero più alto) contiene i dati più significativi. Con l'attivazione di questo collegamento, gli sprite di solito vengono spostati insieme, pur mantenendo capacità di movimento indipendente. La più vasta scelta di colori, tuttavia, è disponibile soltanto nei punti in cui si sovrappongono.
- I bit 6-3 sono riservati per usi futuri (azzerare).
- Il bit 2 contiene il bit V8 della coordinata verticale iniziale.
- Il bit 1 contiene il bit V8 della coordinata verticale finale.
- Il bit 0 contiene il bit H0 della coordinata orizzontale iniziale.

REGISTRI DATI

SPR0DATA e SPR0DATB

Questi registri vengono normalmente utilizzati dai canali DMA degli sprite. Sono i registri che contengono gli ultimi dati trasferiti tramite i canali stessi.

Sommario dei registri di colore degli sprite

Nelle tavole seguenti si vede il modo in cui vengono usate le word di dati nella selezione dei colori.

Tavola 4.6: Registri di colore per gli sprite singoli

Sprite singoli		**Registri di colore**
Sprite	**Valore**	
0 o 1	00	Trasparente
	01	17
	10	18
	11	19
2 o 3	00	Trasparente
	01	21
	10	22
	11	23
4 o 5	00	Trasparente
	01	25
	10	26
	11	27
6 o 7	00	Trasparente
	01	29
	10	30
	11	31

Tavola 4.7: Registri di colore per gli sprite collegati

Sprite collegati	
Valore	**Registro di colore**
0000	trasparente
0001	17
0010	18
0011	19
0100	20
0101	21
0110	22
0111	23
1000	24
1001	25
1010	26
1011	27
1100	28
1101	29
1110	30
1111	31

INTERAZIONI FRA GLI SPRITE E GLI ALTRI OGGETTI VIDEO

I playfield condividono lo schermo con gli sprite. Il Capitolo 7 mostra come stabilire le reciproche priorità e come rilevare eventuali collisioni (sovrapposizioni) tra playfield e sprite.

5 HARDWARE AUDIO

Introduzione

Questo capitolo spiega come accedere direttamente alle risorse audio dell'Amiga nella generazione di suoni. Gli argomenti principali sono:

- Una breve panoramica sulla generazione dei suoni tramite computer.
- La produzione di suoni semplici, continui e variabili, e quindi di suoni più complessi.
- L'uso dei canali audio per effetti speciali, la stereofonia e l'uso di un canale per modularne un altro.
- Come produrre suoni di alta qualità entro i limiti imposti dal sistema.

Un paragrafo al termine del capitolo fornisce l'elenco dei valori da utilizzare per la riproduzione delle note musicali.

Questo capitolo non è certo un trattato sulla generazione dei suoni tramite elaboratore, argomento che richiederebbe uno spazio molto più ampio. Ha invece l'obiettivo di mostrare in che modo si possono sfruttare nel modo migliore le capacità sonore dell'Amiga, evitando tutta la serie di tentativi ed errori a cui spesso ci si deve sottoporre agli inizi.

Per una trattazione più approfondita della sintesi musicale tramite elaboratore, consigliamo i volumi *Introduction to Computer Music* di Wayne A. Bateman (New York, John Wiley & Sons, 1980) e *Musical Applications of Microprocessors* di Hal Chamberlain (Rochelle Park, New Jersey, Hayden, 1980).

INTRODUZIONE ALLA GENERAZIONE DEI SUONI

I suoni viaggiano nell'aria sotto forma di variazioni cicliche della pressione dell'aria, ovverosia onde sonore, fino a raggiungere i padiglioni auricolari. Graficamente, un suono può essere rappresentato tramite un diagramma che mostri la variazione della pressione in funzione del tempo. Le caratteristiche di un suono, così come viene udito, sono in stretta relazione con l'aspetto del grafico. Se la forma d'onda è regolare e periodica, verrà udita come un suono continuo (grave o acuto), ovvero come una nota musicale (ogni ripetizione della forma d'onda viene definita "ciclo").

Se la forma d'onda è irregolare non sarà rilevabile nessuna tonalità precisa, come avviene per l'acqua che scorre. Il numero di ripetizioni della forma d'onda (la sua frequenza) influisce sulla tonalità: suoni con frequenza più alta saranno più acuti. Gli esseri umani possono in genere udire suoni la cui frequenza oscilla tra i 20 e i 20 mila cicli al secondo. L'ampiezza della forma d'onda (il punto più alto del grafico), è invece in relazione con il *volume* del suono. Infine, l'aspetto generale della forma d'onda determina la sua qualità sonora, il *timbro*. La Figura 5.1 mostra un particolare tipo di forma d'onda, detta onda sinusoidale, limitata al primo ciclo.

Gli attributi principali di un suono sono considerati in genere l'ampiezza e la frequenza. La frequenza è costituita dal numero di cicli al secondo, e la più comune unità di misura di questa grandezza è l'Hertz (Hz), che corrisponde a un ciclo al secondo. Valori maggiori possono essere misurati in Kilohertz (KHz) o in Megahertz (MHz).

La frequenza è in stretta relazione con la tonalità apparente di un suono. Quando aumenta l'una cresce anche l'altra, secondo una relazione esponenziale. Un aumento da 100 Hz a 200 Hz aumenta considerevolmente la tonalità del suono, mentre un aumento da 1000 a 1100 Hz si avverte a malapena. La tonalità musicale è rappresentata tramite ottave. Una nota più alta di un'ottava ha una frequenza doppia, e viene percepita con una tonalità doppia.

Il secondo parametro che definisce una forma d'onda è la sua ampiezza. In un circuito elettronico, l'ampiezza dipende dal voltaggio della corrente che attraversa il circuito. Quando il segnale viene inviato a un altoparlante, l'ampiezza viene espressa in watt. L'intensità con la quale il suono viene percepito dall'orecchio si misura invece in decibel (db). L'udito umano ha una capacità di circa 120 db; 1 db è la minima intensità udibile. Un aumento di 10 db corrisponde circa a un volume doppio, e l'aumento di 1 db è il minimo cambiamento avvertibile in un suono

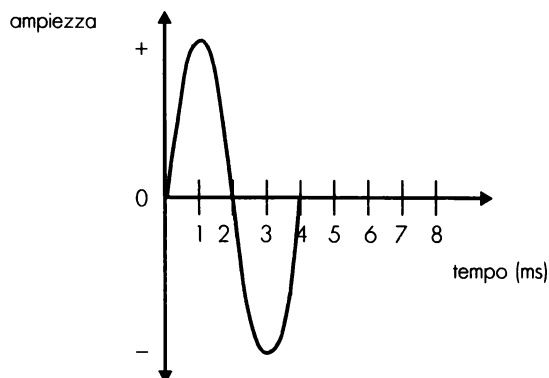



Figura 5.1: Onda sinusoidale

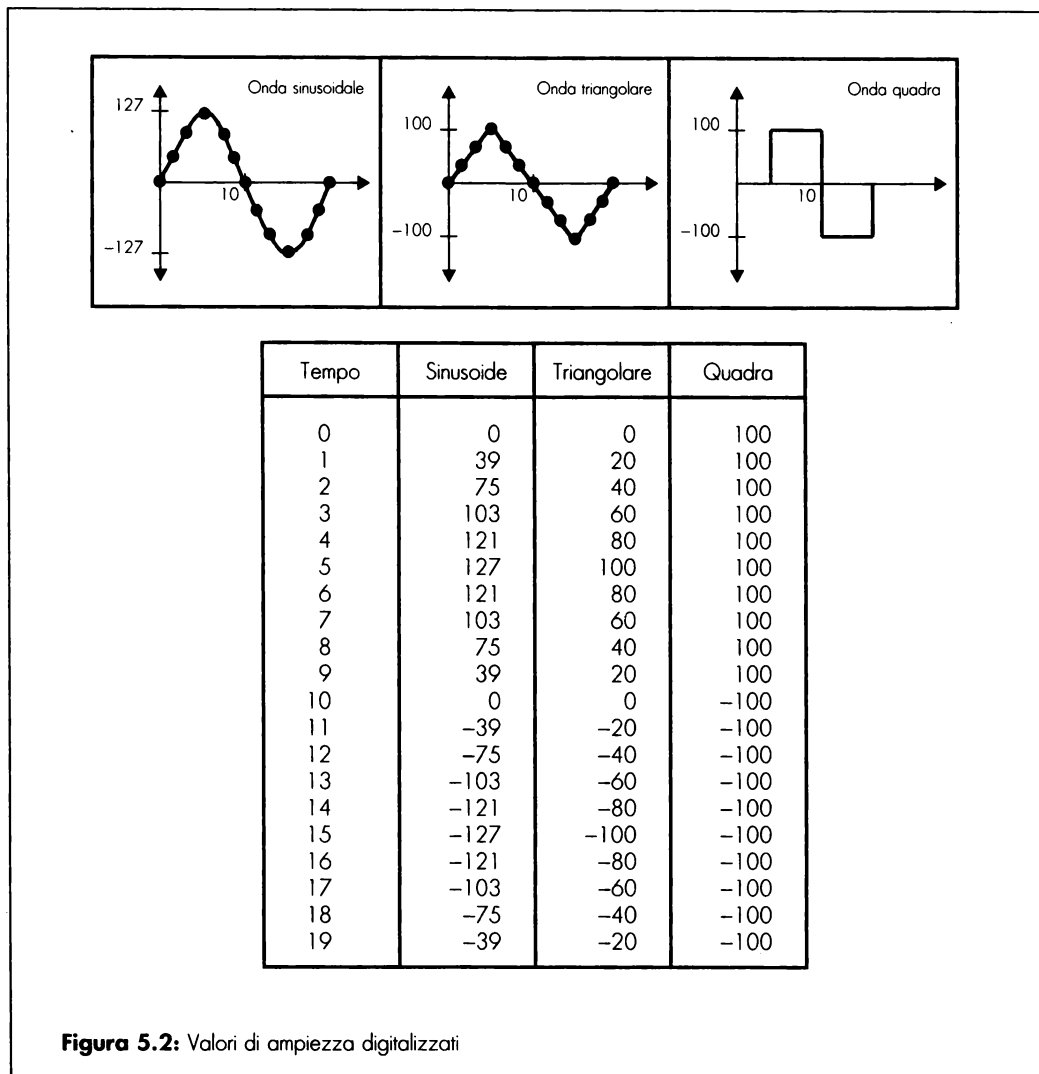
di volume medio. Il volume, che è rappresentato dall'ampiezza del segnale in uscita, ha una corrispondenza logaritmica con il livello dei decibel.

I parametri di ampiezza e di frequenza di un'onda sinusoidale sono completamente indipendenti l'uno dall'altro, ma questo non è vero nella percezione. I suoni a frequenza più bassa, infatti, diminuiscono di volume molto più velocemente di quelli ad alta frequenza.

Il terzo attributo di un suono, il timbro, dipende dalla presenza o dall'assenza di armoniche. Qualsiasi forma d'onda complessa non è altro che la somma di tante onde sinusoidali di diversa ampiezza, frequenza e fase (il punto d'inizio sull'asse dei tempi). Queste onde sinusoidali vengono chiamate armoniche. Un'onda quadra, per esempio, è composta da un numero infinito di armoniche

In sostanza, quindi, tutti i suoni stabili possono essere descritti tramite frequenza, ampiezza, e ampiezze delle relative armoniche. Gli equivalenti "udibili" di questi parametri sono rispettivamente la tonalità, il volume e il timbro. Un suono mutevole può essere considerato

Tempo	Sinusoide	Triangolare	Quadra
0	0	0	100
1	39	20	100
2	75	40	100
3	103	60	100
4	121	80	100
5	127	100	100
6	121	80	100
7	103	60	100
8	75	40	100
9	39	20	100
10	0	0	-100
11	-39	-20	-100
12	-75	-40	-100
13	-103	-60	-100
14	-121	-80	-100
15	-127	-100	-100
16	-121	-80	-100
17	-103	-60	-100
18	-75	-40	-100
19	-39	-20	-100

Figura 5.2: Valori di ampiezza digitalizzati

come un suono stabile i cui parametri cambiano con il tempo.

Nella riproduzione analogica dei suoni, un'apparecchiatura come un registratore a nastro registra le forme d'onda e la loro frequenza tramite una rappresentazione variabile continua della pressione dell'aria. Il registratore in seguito riproduce il suono inviando queste forme d'onda a un amplificatore che le trasforma in una sequenza di variazioni di voltaggio. Le variazioni vengono quindi inviate agli altoparlanti che le trasformano in vibrazioni nella pressione dell'aria che l'ascoltatore percepisce sotto forma di suono.

Un computer, però, non può memorizzare informazioni di tipo analogico. Una forma d'onda dev'essere rappresentata tramite una sequenza finita di numeri. Questa trasformazione viene ottenuta tramite la divisione dell'asse dei tempi in tanti segmenti uguali, ognuno dei quali corrisponde a un intervallo di tempo abbastanza piccolo da rappresentare un'apprezzabile variazione nella forma d'onda. Ognuno dei punti che ne risultano viene definito "campione", o punto di campionamento. Questi campioni vengono immagazzinati sequenzialmente in memoria e possono essere riprodotti alla frequenza con cui sono stati raccolti. Il computer li sottopone a un convertitore digitale-analogico (DAC) che li trasforma in una serie analogica di variazioni di voltaggio. Per produrre il suono, le forme d'onda devono essere inviate a un amplificatore e quindi a un altoparlante.

La Figura 5.2 mostra un'onda sinusoidale, un'onda triangolare e un'onda quadra, insieme a una serie di valori di campionamento. Si noti che le illustrazioni contengono meno punti di quelli elencati nella tabella. I limiti massimi dell'intervallo di campionamento sono −128 e 127.

L'HARDWARE AUDIO DELL'AMIGA

L'Amiga possiede quattro canali audio, ognuno programmabile indipendentemente, che danno la possibilità di produrre complessi effetti sonori. È possibile inoltre collegare i canali in modo che l'uno moduli il suono proveniente dall'altro, o per produrre effetti stereofonici.

Ogni canale comprende un convertitore digitale-analogico controllato da un canale di accesso diretto alla memoria (DMA). Il DMA audio trasferisce al massimo due campionamenti per ogni linea di scansione. Nel caso di semplici suoni periodici, il DMA può automaticamente ripetere la medesima forma d'onda ed è anche possibile programmare effetti sonori più complessi.

Vi sono due metodi per produrre suoni sull'Amiga: generarli automaticamente tramite DMA e generarli direttamente. Con il primo sistema, l'hardware si procura automaticamente i dati audio tramite accesso diretto alla memoria.

Creazione e riproduzione di un suono

Questo paragrafo spiega come creare e riprodurre un semplice suono stabile, e introduce i concetti fondamentali per la creazione musicale tramite l'Amiga.

Ecco i principali passi da seguire:

1. Decidere quale canale utilizzare.

2. Definire la forma d'onda e crearne i dati in memoria.

3. Impostare i registri che informano il sistema sulla posizione dei dati e sulla loro lunghezza.

4. Impostare il volume.

5. Selezionare la frequenza di campionamento, ovvero la velocità con cui verranno riprodotti i dati.

6. Abilitare il DMA del canale prescelto.

SCELTA DEL CANALE

L'Amiga possiede quattro canali audio. I canali 0 e 3 sono collegati al jack di output del canale sinistro. I canali 1 e 2 a quello del canale destro. Si deve selezionare uno dei canali assegnati al lato prescelto.

CREAZIONE DEI DATI DELLA FORMA D'ONDA

La forma d'onda usata come esempio in questo paragrafo è una semplice onda sinusoidale, che produce un tono puro. Per risparmiare memoria, in genere viene memorizzato solo un ciclo completo della forma d'onda. Per ottenere un suono stabile e senza cambiamenti, i valori iniziali e finali e l'inclinazione della curva agli estremi devono essere collegati. Questo assicura che la ripetizione della forma d'onda produca un suono regolare.

I dati del suono sono organizzati come gruppi di valori a otto bit. Ogni valore corrisponde a un punto di campionamento della forma d'onda. Ogni word di dati prelevata per il canale audio consiste di due campioni il cui valore può variare da −128 a +127.

La serie di dati che segue dà origine a una buona approssimazione di un'onda sinusoidale. È da notare che i dati sono memorizzati secondo l'ordine degli indirizzi, con il primo byte all'indirizzo più basso, il secondo in quello successivo e così via. Inoltre, il primo byte di dati deve trovarsi a un indirizzo pari perché il DMA audio preleva una word (16 bit) alla volta, e considera il valore letto come due byte di dati.

Per utilizzare il canale audio 0 bisogna scrivere l'indirizzo iniziale dei dati audio nel registro AUD0LC. Per semplificare, nella tavola abbiamo riunito due registri reali (AUD0LCH e AUD0LCL) nell'unico registro "AUD0LC". Si noti che i dati devono risiedere in un'area di memoria di tipo chip, altrimenti i canali del DMA audio non potrebbero accedervi.

Tavola 5.1: Esempio di valori per l'uso di un canale

AUD0LC	100	98
AUD0LC+2	92	83
AUD0LC+4	71	56
AUD0LC+6	38	20
AUD0LC+8	0	-20
AUD0LC+10	-38	-56
AUD0LC+12	-71	-83
AUD0LC+14	-92	-98
AUD0LC+16	-100	-98
AUD0LC+18	-92	-83
AUD0LC+20	-71	-56
AUD0LC+22	-38	-20
AUD0LC+24	0	20
AUD0LC+26	38	56
AUD0LC+28	71	83
AUD0LC+30	92	98

COME INFORMARE IL SISTEMA SULLA POSIZIONE DEI DATI

Per fornire i dati ai canali audio, il sistema deve sapere dove si trovano e qual è la loro lunghezza in word.

I registri di locazione AUDxLCH e AUDxLCL contengono l'indirizzo iniziale dei dati audio relativi al canale "x". Dal momento che questi due registri sono contigui, scrivere una long word all'indirizzo di AUDxLCH ha l'effetto di inizializzare correttamente anche AUDxLCL. La "x" nel nome del registro sta a indicare il numero del canale audio in questione, e può quindi assumere i valori 0, 1, 2 e 3.

Questi registri sono registri di *locazione* e non di *puntamento*. La differenza consiste nel fatto che è necessario impostarne il contenuto una volta soltanto: non occorre reinizializzare i registri a ogni ciclo per ripetere la stessa forma d'onda. Ogniqualvolta il sistema termina la lettura dei dati, ricorre al contenuto di questi stessi registri per riportare automaticamente la lettura all'inizio. Supponendo che la prima word di dati si trovi all'indirizzo "audiodata", e che si desideri utilizzare il canale 0, ecco come si devono impostare questi registri:

```
LEA     CUSTOM,a0
LEA     audiodata,a1
MOVE.L  a1,AUD0LCH(a0)
```

La lunghezza dei dati è pari alla metà dei punti di campionamento presenti nella forma d'onda, ovvero alla quantità di dati espressa in word. Nell'esempio precedente, la lunghezza risulta di 16 word. Questo dato andrà scritto nel registro "lunghezza dati" del canale corrispondente. Il registro è denominato AUDxLEN, dove "x", come di consueto, sta a indicare il numero del canale. Ecco come s'imposta a 16 il registro AUD0LEN.

```
LEA     CUSTOM,a0
MOVE.W  #16,AUD0LEN(a0)
```

SELEZIONE DEL VOLUME

Con il termine "volume" qui s'intende il volume effettivo del suono generato da un dato canale. Il volume relativo dei suoni, ovvero quello delle singole forme d'onda, coincide con l'ampiezza. Ognuno dei quattro canali audio possiede un registro a 7 bit per il controllo del volume. Questo registro, detto AUDxVOL, va impostato con un valore intero variabile da 0 a 64. Questi valori corrispondono a un determinato livello di decibel. Al termine del capitolo vi è una tavola che mostra il valore in decibel dei 65 possibili livelli di volume.

Un suono emesso a volume 64, con valori dei dati che varino da –128 a 127, genera un voltaggio in uscita oscillante tra +0,4 volt e –0,4 volt. Ecco alcuni livelli di volume e i corrispondenti valori in decibel.

Tavola 5.2: Valori del volume

Volume	**Valore in decibel**	
64	0	Volume massimo
48	-2,5	
32	-6,0	
16	-12,0	12 decibel in meno rispetto al volume massimo

Per impostare il volume di un canale, occorre scriverne il valore nei bit 6-0 di AUDxVOL. Per esempio:

```
LEA     CUSTOM,a0
MOVE.W  #48,AUD0VOL(a0)
```

I decibel vengono indicati come valori negativi a partire da un massimo di 0 perché questo è il modo in cui si indica il livello di registrazione in un registratore a nastro. In genere, questi apparecchi indicano 0 come livello di registrazione ottimale. Qualsiasi valore inferiore viene indicato con numeri negativi.

SELEZIONE DELLA FREQUENZA DI OUTPUT

La tonalità del suono dipende dalla sua frequenza. Per dire al sistema quale frequenza deve usare, occorre specificare il periodo di campionamento. Si tratta cioè di indicare quanti cicli di clock devono trascorrere tra due trasferimenti di un byte di dati al convertitore digitale-analogico. Esiste un registro di periodo per ognuno dei quattro canali audio. Il valore contenuto in questi registri viene usato per una sorta di conto alla rovescia: ogni volta che si raggiunge lo 0, viene trasferito un altro byte di dati. Il periodo viene misurato in cicli di clock per campionamento. Il minimo valore utilizzabile è 123 cicli per campionamento (124 in NTSC) e il massimo è 65535. Per la produzione di suoni di buona qualità, inoltre, vi sono altre restrizioni che saranno trattate in un paragrafo specifico di questo capitolo.
Si tenga presente che più basso è il valore del periodo e più alta è la frequenza.

Limitazioni nella scelta del periodo

Il periodo di campionamento è limitato dal numero di cicli di DMA disponibili per i canali audio. A ogni canale, infatti, viene assegnato uno slot DMA per ogni linea di scansione del video. Quindi, un canale audio può leggere due byte (una word) di dati per ogni linea. Il calcolo seguente fornisce la massima velocità teorica di campionamento.

*2 campioni/linea * 312.5 linee/quadro * 50 quadri/secondo = 31.250 campioni/secondo*

Il numero 31.250 rappresenta però un massimo teorico. L'hardware è disegnato infatti in modo tale da poter gestire al massimo 28.867 campioni al secondo. L'intervallo di clock del sistema è di 281,937 nanosecondi, ovverosia 0,281937 microsecondi. La massima velocità di campionamento (28.867 campioni al secondo) equivale a 34,642 microsecondi per campione (1/28.867 = 0,000034642). La formula per calcolare il periodo è la seguente:

$$\text{Valore del periodo} = \frac{\text{intervallo fra i campioni}}{\text{intervallo di clock}} = \frac{\text{costante di clock}}{\text{campioni al secondo}}$$

Perciò, il minimo valore del periodo si ottiene dividendo 34,642 microsecondi per campione per il numero di microsecondi per ciclo:

$$\text{Periodo minimo} = \frac{\text{34,642 microsecondi/campione}}{\text{0,281937 microsecondi/cicli}} = \text{123 cicli/campione}$$

oppure:

$$\text{Periodo minimo} = \frac{3.546.895 \text{ cicli/secondo}}{28.867 \text{ campioni/secondo}} = 123 \text{ cicli/campione}$$

Pertanto è necessario inserire nei registri di periodo un valore minimo di 123 per essere sicuri che il DMA di sistema abbia il tempo di trasferire il campione successivo. Se viene specificato un periodo inferiore, il DMA audio non ha il tempo di trasferire il dato seguente. Di conseguenza viene riutilizzato lo stesso dato.

28.867 è anche il valore massimo di campionamento per il sistema NTSC; è necessario però utilizzare un valore minimo di 124 cicli/campione.

Valore di clock

	PAL	**NTSC**	**Unità di misura**
Costante di clock	3546895	3579545	cicli al secondo
Intervallo di clock	0,281937	0,279365	microsecondi per ciclo

L'intervallo di clock è l'inverso della costante di clock:

$$\text{intervallo di clock} = \frac{1}{\text{costante di clock}}$$

e in genere viene misurato in microsecondi.

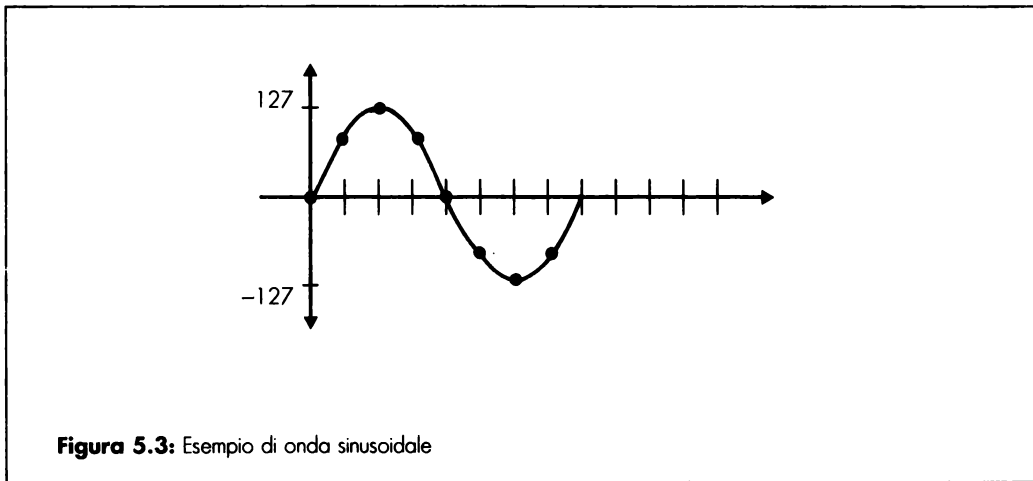
Impostazione del periodo

Dopo aver selezionato l'intervallo, si può calcolare il valore da inserire nei registri tramite la seguente formula:

$$\text{Periodo} = \frac{\text{intervallo desiderato}}{\text{intervallo di clock}} = \frac{\text{costante di clock}}{\text{campioni al secondo}}$$

Come esempio, supponiamo di voler produrre un'onda sinusoidale con la frequenza di 1 KHz, utilizzando otto campioni (quattro word di dati) come in Figura 5.3.

```
Valori:   0
         90
        127
         90
          0
        -90
       -127
        -90
```

Figura 5.3: Esempio di onda sinusoidale

Per far sì che una serie di otto campioni generi un suono alla frequenza di 1 KHz (1000 cicli al secondo), ogni ciclo dev'essere riprodotto in 1/1000 di secondo. Pertanto ogni valore dovrà essere trasferito in 1/8 di questo tempo. Ciò significa 1000 microsecondi per un ciclo completo e quindi 125 microsecondi per campione. Per riprodurre correttamente questa forma d'onda, il periodo dovrà quindi essere:

$$Periodo = \frac{125\ microsecondi/campione}{0{,}281937\ microsecondi/ciclo} = 443\ cicli/campione$$

Per impostare il registro del periodo si deve scrivere l'opportuno valore nel registro AUDxPER. Nel nostro caso:

```
LEA     CUSTOM,a0
MOVE.W  #443,AUD0PER(a0)
```

Per quanto riguarda la relazione tra periodo e tonalità, si consulti la tabella al termine di questo capitolo.

GENERAZIONE DEL SUONO

Dopo avere definito la posizione dei dati, la lunghezza, il volume e il periodo, è ora possibile avviare la generazione del suono abilitando il canale DMA relativo al canale scelto. Una volta avviato, il DMA prosegue fino a che non viene fermato con un ordine esplicito. La forma d'onda viene ripetuta ininterrottamente, generando un tono continuo, e per ogni ripetizione il sistema utilizza gli indirizzi contenuti nei registri di locazione.

Perché i canali DMA audio possano funzionare (o qualsiasi altro canale DMA, se è per questo), dev'essere impostato il bit DMAEN nel registro DMACON. Qualora siano impostati il bit DMAEN e il bit AUDxEN, viene abilitato il DMA relativo al canale "x". Questi bit e il loro significato sono illustrati nella tavola seguente.

Tavola 5.3: DMA e bit di abilitazione dei canali audio

Registro DMACON

Bit	Nome	Funzione
15	SET/CLR	Quando è impostato a 1, imposta tutti i bit di DMACONW per i quali i bit nella posizione corrispondente sono a 1 e lascia invariati tutti gli altri
9	DMAEN	Solo se questo bit è a 1 può avvenire un accesso alla memoria tramite DMA
3	AUD3EN	Abilita il canale audio 3
2	AUD2EN	Abilita il canale audio 2
1	AUD1EN	Abilita il canale audio 1
0	AUD0EN	Abilita il canale audio 0

Per esempio, per abilitare il canale 0, occorre impostare a 1 i bit 9 e 0:

```
LEA     CUSTOM,a0
MOVE.W  #(DMAF_SETCLR!DMAF_AUD0!DMAF_MASTER),DMACON(a0)
```

ARRESTO DEL DMA AUDIO

È possibile "spegnere" un canale in qualunque momento azzerando il corrispondente bit AUDxEN nel registro DMACON. Non è tuttavia possibile riprendere l'output dallo stesso punto semplicemente riportando a 1 il bit. Quando si abilita un canale DMA audio, infatti, il trasferimento dei dati riprende dall'inizio della lista ovvero dall'indirizzo contenuto nei registri di locazione AUDxLCH e AUDxLCL. Soltanto se il canale audio viene disabilitato per un periodo di tempo molto breve (inferiore a due periodi di campionamento) può accadere che continui da dove era stato interrotto.

L'esempio seguente mostra come arrestare un canale DMA audio.

```
LEA     CUSTOM,a0
MOVE.W  #(DMAF_AUD0),DMACON(a0)
```

SOMMARIO

Riassumiamo i passi da seguire per generare un semplice suono costante:

1. Definire la forma d'onda.
2. Creare una serie di dati costituita da coppie di campioni (due per ogni word).
3. Impostare i registri di locazione: AUDxLCH e AUDxLCL.
4. Impostare il registro AUDxLEN a seconda della quantità di dati (espressa in word).
5. Impostare il volume nel registro AUDxVOL.

6. Impostare il registro del periodo, AUDxPER.

7. Abilitare il DMA audio impostando nel registro DMACON il bit 9 e i bit relativi ai canali che si desiderano attivare.

ESEMPIO

In questo esempio, che raccoglie tutti i frammenti di codice già presentati nel corso del capitolo, una semplice onda sinusoidale viene riprodotta tramite il canale 0. Si è supposto di avere un accesso esclusivo all'hardware audio, pertanto l'esempio può non funzionare correttamente in un ambiente multitasking.

```
        LEA     CUSTOM,a0         ;Base dei chip custom
        LEA     DATA(pc),a1       ;Indirizzo dei dati
        MOVE.L  a1,AUD0LCH(a0)    ;Imposta AUD0LCH e AUD0LCL
        MOVE.W  #4,AUD0LEN(a0)    ;Lunghezza in word
        MOVE.W  #64,AUD0VOL(a0)   ;Volume massimo
        MOVE.W  #443,AUD0PER(a0)  ;Frequenza = 1 KHz
        MOVE.W  #(DMAF_SETCLR!DMAF_AUD0!DMAF_MASTER),DMACON(a0)
        RTS                       ;Ritorna al programma principale

        CNOP    0,2               ;Allinea a un indirizzo pari
DATA:
        DC.B    0,90,127,90,0,-90,-127,-90

        END
```

Produzione di suoni complessi

Finora abbiamo parlato soltanto di suoni semplici, ma è possibile creare anche suoni complessi, ottenuti per esempio unendo due suoni, oppure fondendo in una singola voce diverse note musicali oppure modulando i suoni.

UNIONE DI SUONI

I suoni si possono unire utilizzando opportunamente i registri di locazione e lunghezza, l'abilitazione del DMA, e la riscrittura degli stessi registri in preparazione della successiva forma d'onda che si desidera unire alla prima. Il compito è reso più semplice dalla sincronizzazione ottenibile mediante gli interrupt e dall'esistenza di registri di backup. I registri di locazione e di lunghezza vengono letti dal DMA audio all'inizio delle operazioni di output. I canali DMA, in seguito, memorizzano questi dati in registri interni di backup. Una volta che il contenuto originario dei registri è stato letto dai canali DMA audio, è possibile modificarne il valore senza che ciò influisca sull'operazione in corso. È quindi possibile impostare questi registri, attivare il DMA, e poi riscriverli in preparazione della forma d'onda successiva.

Gli interrupt hanno luogo immediatamente dopo che il canale DMA audio ha letto i registri di locazione e di lunghezza e li ha immagazzinati nei registri di backup, e sfruttando l'interrupt è possibile riscrivere i registri.

Questa combinazione di interrupt e registri di backup consente al programma di mantenersi sempre un passo più avanti del canale DMA audio, e garantisce un output continuo e regolare.

Se i registri non vengono riscritti, viene automaticamente ripetuta la medesima forma d'onda. Ogni volta che il contatore della lunghezza raggiunge lo zero, entrambi i registri (locazione e lunghezza) vengono reimpostati con i medesimi valori e la generazione del suono continua senza interruzioni.

Esempio

Si supponga di voler unire un'onda sinusoidale e un'onda triangolare, suonandole alternativamente. La seguente sequenza mostra quello che deve fare il programma e illustra le sue interazioni con il sistema DMA audio. Nell'esempio si presume che periodo, volume e lunghezza dei dati rimangano gli stessi per le due forme d'onda.

Codice di interrupt

Se (onda = triangolare)
impostare AUD0LC con l'indirizzo dell'onda sinusoidale.

Altrimenti, se (onda = sinusoidale)
impostare AUD0LC con l'indirizzo dell'onda triangolare.

Programma principale

1. Impostare volume, periodo e lunghezza.
2. Impostare AUD0LC con l'indirizzo dell'onda sinusoidale.
3. Attivare il DMA.
4. Continuare con il resto del programma.

Azioni del sistema

Non appena viene attivato il DMA:

1. Copia la lunghezza dei dati da AUD0LEN e la salva in un registro di backup.
2. Copia l'indirizzo dei dati da AUD0LC e la salva in un registro di backup (da utilizzare come puntatore ai dati).
3. Genera una richiesta di interrupt del 68000.
4. Inizia il trasferimento dei dati tramite DMA.

PRODUZIONE DI PIÙ SUONI NELLO STESSO ISTANTE

Per produrre più di un suono nello stesso istante si può per esempio utilizzare più di un canale audio, oppure si possono sommare i dati relativi a più suoni e riprodurli tramite un unico canale.
Dal momento che ogni canale è programmabile indipendentemente dagli altri, ciascuno possiede una propria serie di dati e può quindi generare una nota diversa.

MODULAZIONE DEI SUONI

Per ottenere effetti audio più complessi si può utilizzare un canale per modularne un altro. Ciò aumenta la gamma delle possibili sonorità. La modulazione può avvenire in frequenza, in ampiezza o in frequenza e in ampiezza contemporaneamente.
La modulazione d'ampiezza influisce sul volume del suono. Viene in genere utilizzata per produrre effetti di vibrato o di tremolo. La modulazione di frequenza influisce invece sul periodo del suono. Sebbene la forma d'onda rimanga sempre la stessa, la tonalità risulta alzata o abbassata.
Perché il sistema utilizzi un canale per modularne un altro, i due canali devono essere collegati. I suoni prodotti dipendono poi da come sono impostati i bit di collegamento presenti nel registro ADKCON. Se un canale è attivo produce suoni, ma se è attivato uno dei suoi bit di collegamento allora il canale serve esclusivamente per modulare il suono proveniente dal canale con il numero immediatamente superiore: le word di dati a esso associate non vengono più interpretate come due singoli byte, ma come word "di modulazione". Queste word vengono inserite negli opportuni registri del canale modulato ogni volta che il conteggio relativo al periodo del canale modulatore raggiunge lo zero.
Per modulare soltanto l'ampiezza, queste word devono contenere informazioni riguardanti il volume, secondo il seguente formato:

Bit	**Funzione**
15-7	Non utilizzati
6-0	Informazioni sul volume, V6-V0

Per modulare soltanto la frequenza, queste word devono invece contenere informazioni riguardanti il periodo, secondo il seguente formato:

Bit	**Funzione**
15-0	Informazioni sul periodo, P15-P0

Volendo modulare sia il periodo che il volume del canale, è necessario specificare entrambe le funzioni. Per esempio, se il canale 0 viene utilizzato per modulare periodo e volume del canale 1, bisogna impostare due bit del registro ADKCON: il bit 0 per la modulazione di volume, e il bit 4 per la modulazione del periodo. Quando periodo e volume vengono entrambi modulati, le word di dati del canale modulatore vengono interpretate una volta come informazioni sul volume e una volta come informazioni sul periodo.
La serie di dati riportata nella Tavola 5.4 mostra il diverso modo d'interpretare i dati quando un canale viene usato per la generazione di suoni e quando viene usato per modulare il volume, il periodo, o entrambi.

Tavola 5.4: Interpretazione dei dati in modo collegato

Word di dati	**Modulazione Indipendente**	**Modulazione periodo e volume**	**Solo periodo**	**Solo volume**
Word 1	dato dato	volume dell'altro canale	periodo	volume
Word 2	dato dato	periodo dell'altro canale	periodo	volume
Word 3	dato dato	volume dell'altro canale	periodo	volume
Word 4	dato dato	periodo dell'altro canale	periodo	volume

Le lunghezze dei dati del canale modulatore e del canale modulato sono completamente indipendenti.

I canali vengono collegati dal sistema secondo un ordine predeterminato, come mostra la Tavola 5.5. Per collegare un canale a un altro è sufficiente porre a 1 il corrispondente bit di collegamento. Facendo ciò, si disabilita la generazione del suono da parte del canale modulatore. Poiché un canale collegato influisce sempre sul canale immediatamente superiore, il canale 3 non può modularne nessun altro: impostandone a 1 i bit di collegamento si ottiene soltanto di disabilitare l'output audio.

Tavola 5.5: Collegamento dei canali nella modulazione

Registro ADKCON

Bit	**Nome**	**Funzione**
7	ATPER3	Utilizza il canale audio 3, quindi non modula nulla
6	ATPER2	Utilizza il canale audio 2 per modulare il periodo del canale 3
5	ATPER1	Utilizza il canale audio 1 per modulare il periodo del canale 2
4	ATPER0	Utilizza il canale audio 0 per modulare il periodo del canale 1
3	ATVOL3	Utilizza il canale audio 3, quindi non modula nulla
2	ATVOL2	Utilizza il canale audio 2 per modulare il volume del canale 3
1	ATVOL1	Utilizza il canale audio 1 per modulare il volume del canale 2
0	ATVOL0	Utilizza il canale audio 0 per modulare il volume del canale 1

Generazione di suoni di alta qualità

Volendo produrre suoni di elevata qualità è necessario prendere in considerazione i seguenti elementi:

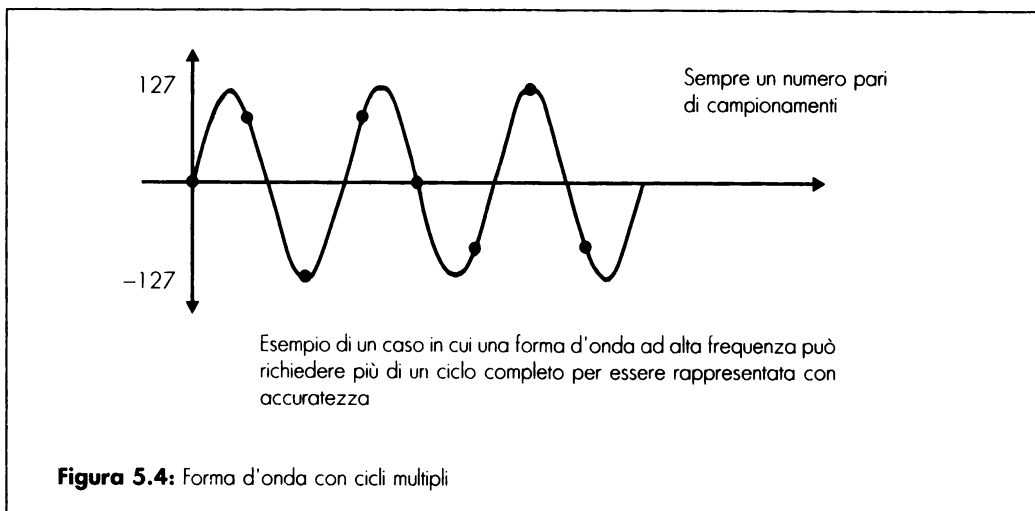
- Transizioni della forma d'onda.
- Frequenza di campionamento.
- Efficienza.
- Riduzione del rumore.
- Distorsioni di aliasing.
- Limiti del filtro passa-basso.

TRANSIZIONI DELLA FORMA D'ONDA

Per evitare fastidiosi rumori nelle transizioni da una forma d'onda all'altra, è necessario "addolcirle". A questo scopo basta assicurarsi che il punto iniziale e quello finale della forma d'onda abbiano all'incirca lo stesso valore, che questo valore sia il più vicino possibile allo zero, e infine che le ampiezze medie di ogni onda abbiano all'incirca il medesimo valore. L'ampiezza media di una forma d'onda è data dalla somma dei valori dei byte che la compongono divisa per il loro numero.

FREQUENZA DI CAMPIONAMENTO

Se è necessaria una notevole precisione di frequenza, può accadere che la frequenza desiderata cada proprio tra due valori assegnabili al periodo ma consecutivi, senza essere

Figura 5.4: Forma d'onda con cicli multipli

abbastanza vicina a nessuno dei due. In questi casi può essere necessario modificare leggermente la lunghezza dei dati, oltre che variare il periodo.

Per frequenze più alte, può anche essere necessario utilizzare tavole di dati che contengano più di un ciclo completo della forma d'onda per riprodurre accuratamente la frequenza desiderata (si veda la Figura 5.4).

EFFICIENZA

Nella gestione del DMA audio sono coinvolte parecchie operazioni. Se si desidera ottenere una sintesi sonora del massimo livello raggiungibile, si devono evitare (per quanto è possibile) operazioni di controllo da parte del sistema. Come regola generale, con l'aumentare delle dimensioni dei buffer di dati, il sistema è sempre meno costretto a generare interrupt per la nuova impostazione dei puntatori, e di conseguenza diminuisce la frequenza dei suoi interventi. Utilizzando un'unica forma d'onda, l'hardware reinizializza automaticamente i puntatori, per cui l'intervento del sistema si riduce a zero.

Abbiamo visto come si uniscono fra loro più suoni, modificando i registri di locazione e sincronizzando l'operazione tramite gli interrupt. Se però il sistema è molto occupato in altri compiti, è possibile che la risposta agli interrupt non avvenga abbastanza in fretta da assicurare una transizione regolare da un suono al successivo. È perciò consigliabile utilizzare una tavola di dati più lunga possibile, per sfruttare maggiormente le possibilità offerte dal DMA e minimizzare al tempo stesso il numero di interrupt ai quali il 68000 deve rispondere.

RIDUZIONE DEL RUMORE

Per ridurre il livello di rumore e produrre un suono pulito, è opportuno utilizzare l'intera gamma da -128 a +127 nella rappresentazione di una forma d'onda. Questo aumenta il numero di bit di precisione e riduce il rumore (errore di quantizzazione) che "intorbida" il segnale. Il rumore è dovuto agli errori che si introducono con gli arrotondamenti. Cercando di riprodurre un segnale, per esempio un'onda sinusoidale, la precisione con cui si può rappresentare l'ampiezza dei campioni è limitata. La differenza tra il valore reale e questo numero costituisce l'errore di arrotondamento.

Raddoppiando l'ampiezza viene ovviamente raddoppiata anche l'accuratezza, dimezzando quindi il livello del rumore.

In altre parole, cercare di rappresentare una forma d'onda tramite valori che vadano da -3 a +3, dà come risultato un errore di arrotondamento assai maggiore che non utilizzando l'intervallo tra -128 e +127. In proporzione, infatti, il valore usato per rappresentare l'ampiezza ha un errore minore. Aumentando il numero dei possibili valori di campionamento diminuisce la dimensione relativa degli intervalli di divisione, e, pertanto, quella dell'errore.

Anche per produrre suoni di bassa intensità conviene utilizzare l'intero intervallo di ampiezze disponibile, intervenendo invece sui controlli del volume. In questo modo viene mantenuto sempre il medesimo rapporto fra segnale e rumore.

DISTORSIONE DI ALIASING

Quando si riproduce una forma d'onda può verificarsi una distorsione se la frequenza di campionamento "batte" (ovvero si combina) con quella da riprodurre. Ciò dà origine a due frequenze addizionali, una costituita dalla somma delle due frequenze, l'altra dalla loro differenza. Questo fenomeno è chiamato distorsione di aliasing o "dei battimenti".

La distorsione di aliasing scompare quando la frequenza di campionamento supera almeno di 7 KHz la frequenza di output. Ciò pone le frequenze di battimento all'esterno del dominio del filtro passa-basso, e le elimina. La Figura 5.5 rappresenta graficamente il dominio del filtro passa-basso usato dal sistema. La Figura 5.6 mostra invece come si può utilizzare una frequenza di campionamento di 12 KHz per riprodurre un'onda di 4 KHz. Entrambe le frequenze di battimento sono infatti all'esterno del dominio del filtro:

12 + 4 = 16 KHz

12 - 4 = 8 KHz

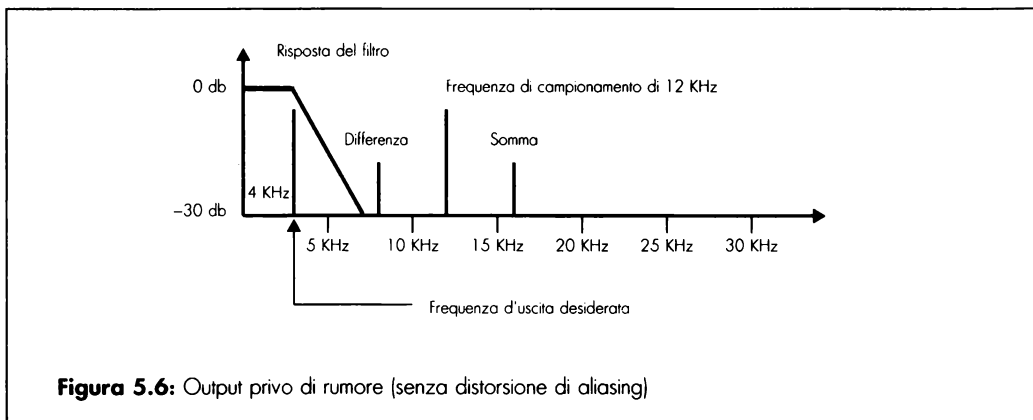
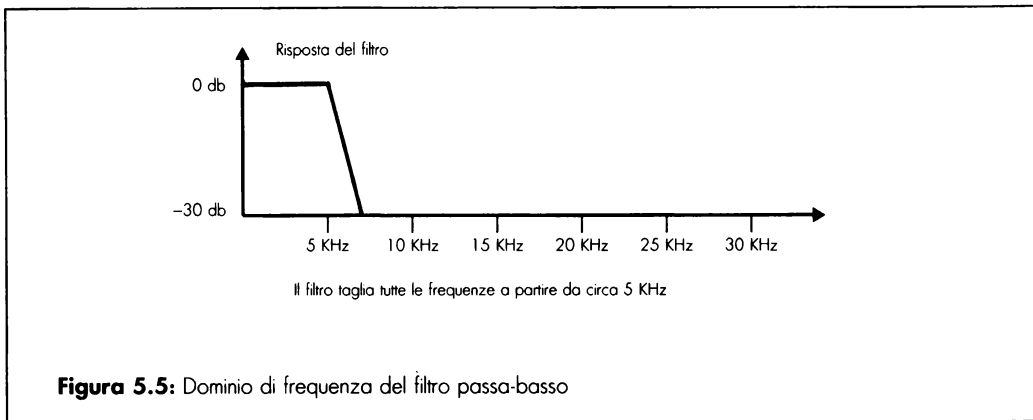
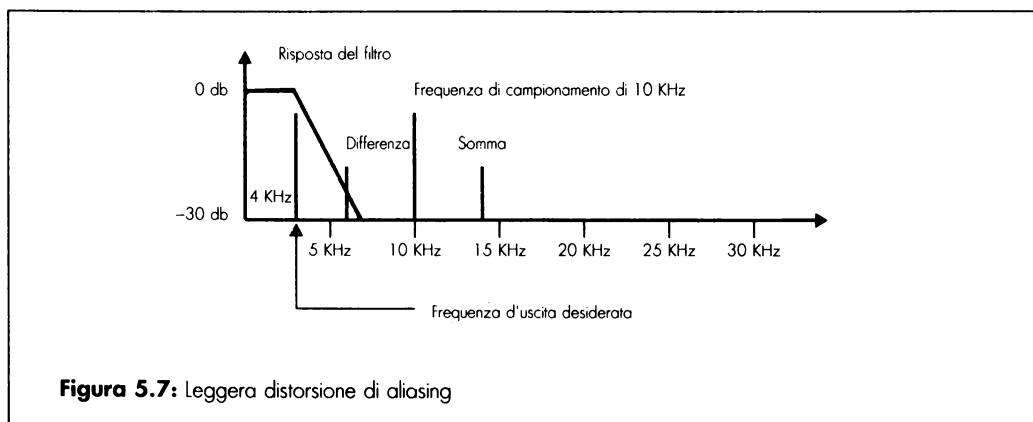
La Figura 5.7 mostra come sia invece inaccettabile una frequenza di campionamento di 10 KHz per riprodurre un'onda di 4 KHz. Una delle frequenze di battimento, infatti, (10 - 4) si trova all'interno del dominio del filtro, e produce frequenze indesiderate nel suono finale.

Queste considerazioni danno origine all'equazione seguente, che riassume la necessità di avere una frequenza di campionamento superiore di almeno 7 KHz rispetto alla frequenza di output, in modo che entrambe le frequenze di battimento siano eliminate dal filtro passa-basso:

minima frequenza di campionamento = massima frequenza componente + 7 KHz

Figura 5.5: Dominio di frequenza del filtro passa-basso

Figura 5.6: Output privo di rumore (senza distorsione di aliasing)

Figura 5.7: Leggera distorsione di aliasing

La frequenza che compare nell'equazione è stata denominata "massima frequenza componente" poiché la frequenza da riprodurre potrebbe in realtà appartenere a una forma d'onda complessa, costituita da più elementi.

IL FILTRO PASSA-BASSO

Il sistema comprende un filtro passa-basso in grado, come abbiamo visto in precedenza, di eliminare la distorsione di aliasing. Il filtro diventa attivo intorno ai 4 KHz e gradualmente comincia ad attenuare il segnale. In linea generale, frequenze superiori a 7 KHz non sono più udibili chiaramente. Pertanto, la risposta più precisa si ottiene nell'intervallo 0-7 KHz. Generando frequenze in questo intervallo, si dovrebbe selezionare una frequenza di campionamento non inferiore ai 14 KHz, che corrisponde a un periodo variabile da 123 a 256.
Con un periodo superiore a 320, si cominciano a perdere le frequenze più alte tra 0 e 7 KHz, come si vede nella tavola seguente.

Tavola 5.6: Relazione tra periodo e frequenza

	Periodo	**Frequenza camp. (KHz)**	**Frequenza di output (KHz)**
Massima frequenza di campionamento	123	29	7
Minima frequenza di campionamento per un output di 7 KHz	256	14	7
Frequenza di campionamento troppo bassa per un output di 7 KHz	320	11	4

In quasi tutti i modelli di Amiga 2000 e Amiga 500 è presente un bit di controllo che permette all'output audio di evitare il filtro passa-basso. Questo bit è lo stesso bit di output di uno dei

chip 8520 CIA che controlla la luminosità del LED di accensione ("power"). Aggirare il filtro può essere utile nel caso di alcune applicazioni musicali, ma spesso questa tecnica rende necessario ricorrere a un filtro esterno con un'appropriata frequenza di taglio.

Uso diretto (non DMA) dell'audio

È possibile generare un suono scrivendone i dati direttamente nei registri audio una word per volta, anziché predisporre un elenco di dati in memoria. Questo metodo di controllo sfrutta molto il microprocessore e pertanto non è raccomandato.

Comunque, se si vuole seguire questa via basta non abilitare il DMA del canale che s'intende usare, modificando così la temporizzazione degli interrupt. In genere gli interrupt si verificano quando sono stati letti l'indirizzo e la lunghezza dei dati. Nell'uso diretto dei canali audio, invece, gli interrupt si verificano immediatamente dopo l'invio di una word di dati.

Nell'uso diretto, se non vengono impostati nuovi valori dei dati prima che siano trascorsi due intervalli di campionamento, l'output audio non subisce mutamenti: l'ultimo valore continua a essere sottoposto al convertitore digitale-analogico.

I registri di periodo e di volume vengono impostati come di consueto.

La scala musicale temperata

La Tavola 5.7 fornisce una buona approssimazione di una scala temperata di un'ottava (cioè una scala in cui tutti i semitoni sono uguali) tramite un campione di 16 byte. Le colonne "periodo", forniscono il valore da inserire nel registro del periodo. Il registro AUDxLEN deve invece essere impostato a 8. Il campione rappresenta un ciclo della forma d'onda.

Tavola 5.7: Scala temperata, ottenuta con un campione di 16 byte

Periodo PAL	Periodo NTSC	Nota	Freq. ideale	Freq. reale PAL	Freq. reale NTSC
252	254	LA	880,0	879,7	880,8
238	240	LA#	932,3	931,4	932,2
224	226	SI	987,8	989,6	989,9
212	214	DO	1046,5	1045,7	1045,4
200	202	DO#	1108,7	1108,4	1107,5
189	190	RE	1174,7	1172,9	1177,5
178	180	RE#	1244,5	1245,4	1242,9
168	170	MI	1318,5	1319,5	1316,0
159	160	FA	1396,9	1394,2	1398,3
150	151	FA#	1480,0	1477,9	1481,6
141	143	SOL	1568,0	1572,2	1564,5
133	135	SOL#	1661,2	1666,8	1657,2

La tavola fornisce i valori del periodo, da utilizzare con un campione di 16 byte, per ottenere le note della seconda ottava sopra il do centrale. Per ottenere suoni di ottave più basse vi sono due metodi differenti: raddoppiare il valore del periodo o raddoppiare la lunghezza del campione.

Raddoppiando il periodo, il tempo fra ogni valore di campionamento viene raddoppiato, cosicché occorre un tempo doppio per suonare l'intera serie. Ciò significa che la frequenza del suono generato viene dimezzata, il che equivale a un'ottava inferiore. Pertanto, un do riprodotto tramite un periodo di 212 darà origine al do dell'ottava inferiore se suonato con periodo 424.

Allo stesso modo, raddoppiando la lunghezza dei dati, occorrerà un tempo doppio per riprodurre la forma d'onda, e la frequenza della nota generata corrisponderà quindi all'ottava inferiore. Pertanto, avendo due serie di campioni che rappresentano la stessa forma d'onda (una di 8 byte e l'altra di 16 byte) la serie di 16 byte sarà più bassa di un'ottava.

Una serie di campioni rappresenta in genere una singola nota. Per evitare distorsioni di aliasing si dovrebbero utilizzare valori del periodo compresi nell'intervallo 123-256. Ciò corrisponde a una frequenza di riproduzione che va da 14 a 28 mila campioni al secondo (il che rende più efficace il filtro a 7 KHz nell'eliminazione del rumore). Per rimanere entro questi limiti, è necessaria una diversa serie di campioni per ogni ottava.

Se questo non è possibile, diventa necessario utilizzare valori del periodo che spazino in tutto l'intervallo 123-65536. In fase di programmazione quest'operazione è la più semplice, ma può produrre del rumore indesiderato ad alta frequenza.

I valori della Tavola 5.7 sono stati ricavati utilizzando la seguente formula:

$$Frequenza = \frac{costante\ di\ clock}{byte * periodo} = \frac{3546895}{16 * 252} = 879{,}7\ Hz$$

La costante di clock in un sistema PAL vale 3546895 cicli al secondo. In un sistema NTSC è invece di 3579545 cicli al secondo. Questo valore si ottiene dividendo per due il clock di sistema, quindi può cambiare nel caso che si utilizzi un clock esterno, come per esempio con un genlock.

Usando la formula precedente, si possono ricavere i valori del periodo necessari per riprodurre qualunque campione alla frequenza desiderata. La Tavola 5.8 fornisce la miglior approssimazione ottenibile di una scala temperata di cinque ottave ottenuta con cinque campioni diversi. I valori sono stati ricavati tramite la formula sopra riportata. Si noti che in ogni ottava i valori del periodo sono gli stessi, ma viene dimezzata la dimensione del campione. L'esempio riguarda una semplice onda triangolare.

Tavola 5.8: Scala musicale su cinque ottave

Per. PAL	Per. NTSC	Nota	Freq. ideale	Freq. reale PAL	Freq. reale NTSC
252	254	LA	55,00	54,98	55,05
238	240	LA#	58,27	58,21	58,26
224	226	SI	61,73	61,85	61,87
212	214	DO	65,40	65,35	65,34
200	202	DO#	69,29	69,27	69,22
189	190	RE	73,41	73,30	73,59
178	180	RE#	77,78	77,83	77,68
168	170	MI	82,40	82,47	82,25
159	160	FA	87,30	87,13	87,39
150	151	FA#	92,49	92,36	92,60
141	143	SOL	98,00	98,26	97,78
133	135	SOL#	103.82	104,17	103,57

Dimensioni del campione = 256 byte, AUDxLEN = 128

252	254	LA	110,00	109,96	110,10
238	240	LA#	116,54	116,43	116,52
224	226	SI	123,47	123,70	123,74
212	214	DO	130,81	130,71	130,68
200	202	DO#	138,59	138,55	138,44
189	190	RE	146,83	146,61	147,18
178	180	RE#	155,56	155,67	155,36
168	170	MI	164,81	164,94	164,50
159	160	FA	174,61	174,27	174,78
150	151	FA#	184,99	184,73	185,20
141	143	SOL	196,00	196,52	195,56
133	135	SOL#	207,65	208,35	207,15

Dimensioni del campione = 128 byte, AUDxLEN = 64

252	254	LA	220,00	219,92	220,20
238	240	LA#	233,08	232,86	233,04
224	226	SI	246,94	247,41	247,48
212	214	DO	261,63	261,42	261,36
200	202	DO#	277,18	277,10	276,88
189	190	RE	293,66	293,23	294,37
178	180	RE#	311,13	311,35	310,72
168	170	MI	329,63	329,88	329,00
159	160	FA	349,23	348,55	349,56
150	151	FA#	369,99	369,47	370,40
141	143	SOL	392,00	393,05	391,12
133	135	SOL#	415,30	416,70	414,30

Dimensioni del campione = 64 byte, AUDxLEN = 32

252	254	LA	440,00	439,8	440,40
238	240	LA#	466,16	465,72	466,09
224	226	SI	493,88	494,82	494,96
212	214	DO	523,25	522,83	522,71
200	202	DO#	554,37	554,20	553,77
189	190	RE	587,33	586,46	588,74
178	180	RE#	622,25	622,70	621,45
168	170	MI	659,26	659,76	658,00
159	160	FA	698,46	697,11	699,13
150	151	FA#	739,99	738,94	740,80
141	143	SOL	783,99	786,10	782,24
133	135	SOL#	830,61	833,39	828,60

Dimensioni del campione = 32 byte, AUDxLEN = 16

252	254	LA	880,0	879,7	880,8
238	240	LA#	932,3	931,4	932,2

224	226	SI	987,8	989,6	989,9
212	214	DO	1046,5	1045,7	1045,4
200	202	DO#	1108,7	1108,4	1107,5
189	190	RE	1174,7	1172,9	1177,5
178	180	RE#	1244,5	1245,4	1242,9
168	170	MI	1318,5	1319,5	1316,0
159	160	FA	1396,9	1394,2	1398,3
150	151	FA#	1480,0	1477,9	1481,6
141	143	SOL	1568,0	1572,2	1564,5
133	135	SOL#	1661,2	1666,8	1657,2

Dimensioni del campione = 16 byte, AUDxLEN = 8

Campione da 256 byte

0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
32	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62
64	66	68	70	72	74	76	78	80	82	84	86	88	90	92	94
96	98	100	102	104	106	108	110	112	114	116	118	120	122	124	126
127	126	124	122	120	118	116	114	112	110	108	106	104	102	100	98
96	94	92	90	88	86	84	82	80	78	76	74	72	70	68	66
64	62	60	58	56	54	52	50	48	46	44	42	40	38	36	34
32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2
0	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20	-22	-24	-26	-28	-30
-32	-34	-36	-38	-40	-42	-44	-46	-48	-50	-52	-54	-56	-58	-60	-62
-64	-66	-68	-70	-72	-74	-76	-78	-80	-82	-84	-86	-88	-90	-92	-94
-96	-98	-100	-102	-104	-106	-108	-110	-112	-114	-116	-118	-120	-122	-124	-126
-128	-126	-124	-122	-120	-118	-116	-114	-112	-110	-108	-106	-104	-102	-100	-98
-96	-94	-92	-90	-88	-86	-84	-82	-80	-78	-76	-74	-72	-70	-68	-66
-64	-62	-60	-58	-56	-54	-52	-50	-48	-46	-44	-42	-40	-38	-36	-34
-32	-30	-28	-26	-24	-22	-20	-18	-16	-14	-12	-10	-8	-6	-4	-2

Campione da 128 byte

0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
64	68	72	76	80	84	88	92	96	100	104	108	112	116	120	124
127	124	120	116	112	108	104	100	96	92	88	84	80	76	72	68
64	60	56	52	48	44	40	36	32	28	24	20	16	12	8	4
0	-4	-8	-12	-16	-20	-24	-28	-32	-36	-40	-44	-48	-52	-56	-60
-64	-68	-72	-76	-80	-84	-88	-92	-96	-100	-104	-108	-112	-116	-120	-124
-128	-124	-120	-116	-112	-108	-104	-100	-96	-92	-88	-84	-80	-76	-72	-68
-64	-60	-56	-52	-48	-44	-40	-36	-32	-28	-24	-20	-16	-12	-8	-4

Campione da 64 byte

0	8	16	24	32	40	48	56	64	72	80	88	96	104	112	120
127	120	112	104	96	88	80	72	64	56	48	40	32	24	16	8
0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80	-88	-96	-104	-112	-120
-128	-120	-112	-104	-96	-88	-80	-72	-64	-56	-48	-40	-32	-24	-16	-8

Campione da 32 byte

0	16	32	48	64	80	96	112	127	112	96	80	64	48	32	16
0	−16	−32	−48	−64	−80	−96	−112	−128	−112	−96	−80	−64	−48	−32	−16

Campione da 16 byte

0	32	64	96	127	96	64	32	0	−32	−64	−96	−128	−96	−64	−32

Valore del volume in decibel

La Tavola 5.9 fornisce i valori in decibel corrispondenti ai valori di volume disponibili sull'Amiga.

Tavola 5.9: Rapporto tra volume e decibel

Volume	Decibel	Volume	Decibel
64	0,0	32	−6,0
63	−0,1	31	−6,3
62	−0,3	30	−6,6
61	−0,4	29	−6,9
60	−0,6	28	−7,2
59	−0,7	27	−7,5
58	−0,9	26	−7,8
57	−1,0	25	−8,2
56	−1,2	24	−8,5
55	−1,3	23	−8,9
54	−1,5	22	−9,3
53	−1,6	21	−9,7
52	−1,8	20	−10,1
51	−2,0	19	−10,5
50	−2,1	18	−11,0
49	−2,3	17	−11,5
48	−2,5	16	−12,0
47	−2,7	15	−12,6
46	−2,9	14	−13,2
45	−3,1	13	−13,8
44	−3,3	12	−14,5
43	−3,5	11	−15,3
42	−3,7	10	−16,1
41	−3,9	9	−17,0
40	−4,1	8	−18,1
39	−4,3	7	−19,2
38	−4,5	6	−20,6
37	−4,8	5	−22,1
36	−5,0	4	−24,1
35	−5,2	3	−26,6

34	-5,5	2	-30,1
33	-5,8	1	-36,1
		0	- ∞

L'Audio State Machine

La Figura 5.8 cerca d'illustrare tutti gli stati possibili dell'hardware audio (la figura riguarda comunque un solo canale). Gli otto stati possibili del canale sono regolati da un clock costante (3,55 MHz PAL). Tre di essi sono sostanzialmente inutilizzati e costituiscono un passaggio intermedio verso lo stato inerte (000). Dallo stato inerte si possono imboccare due strade: una riguarda le operazioni gestite tramite interrupt (con i dati forniti dal 68000), e l'altra riguarda le operazioni gestite tramite DMA (con i dati forniti dal chip Agnus).

Nelle operazioni gestite tramite interrupt, la transizione al loop principale (stati 010 e 011) avviene immediatamente dopo il trasferimento dei dati da parte del 68000. Nello stato 010 si verifica l'output del byte superiore, nello stato 011 l'output del byte inferiore. La transizione 010→011→010 avviene ogni volta che il contatore di periodo raggiunge il valore 1. Finché gli interrupt vengono soddisfatti in tempo dal 68000 si rimane in questo loop. Altrimenti si rientra nello stato inerte. Viene generato un interrupt per ogni word (011→010).

Nelle operazioni gestite tramite DMA si passa allo stato 001, e le richieste di DMA vengono inoltrate ad Agnus non appena il DMA viene abilitato. A causa dello schema pipe-line di Agnus, la prima word di dati dev'essere ignorata. Non appena questa word di dati arriva, si entra nello stato 101; la richiesta della word successiva è già stata trasmessa. Quando giunge il nuovo dato, avviene la transizione allo stato 010, e il loop principale continua fino a che il DMA non viene disabilitato. Il contatore di lunghezza viene decrementato di 1 a ogni word di dati. Quando raggiunge il valore 1, oltre alla normale richiesta di dati Agnus riceve anche la richiesta di reinizializzare il DMA. Agnus riporta quindi i puntatori al punto di partenza. Inoltre viene ricaricato anche il contatore di lunghezza, e non appena comincia l'output dell'ultima word di dati, viene inviata una richiesta di interrupt al 68000.

Le richieste riguardanti il DMA vengono inviate ad Agnus una volta per ogni linea di scansione, e i dati giungono circa 14 cicli di clock più tardi.

Nel modo collegato le cose sono leggermente diverse. Quando viene collegato il volume, le richieste avvengono come durante le normali operazioni, con la transizione 011→010. Nel collegamento del periodo, invece, avvengono con la transizione 010→011. Se vengono collegati sia il periodo che il volume, le richieste sono inoltrate in entrambi i casi.

Se la frequenza di campionamento è molto più alta del normale massimo (vale a dire circa 29 KHz), i due valori presenti nel buffer vengono ripetuti. Se viene disattivato il filtro passa-basso e il volume è al massimo ($40), questa caratteristica può essere utilizzata per generare portanti modulate fino a 1,79 MHz. La modulazione viene memorizzata in RAM con valori positivi nei byte pari e negativi nei byte dispari.

I simboli utilizzati nel diagramma di stato sono spiegati nella lista seguente. I nomi in maiuscolo indicano segnali esterni, quelli in minuscolo segnali locali della circuiteria audio.

AUDxON	DMA attivato (segnale proveniente da DMACON).
AUDxIP	Interrupt audio in corso (input dai circuiti di interrupt).
AUDxIR	Richiesta di interrupt (output dal canale ai circuiti di interrupt).

intreq1	Richiesta di interrupt che si combina con intreq2 per formare AUDxIR.
intreq2	Prepara la richiesta di interrupt. La richiesta ha luogo dopo la successiva transizione 011→010, nel corso delle normali operazioni.
AUDxDAT	Segnale di caricamento dati. Vengono caricati 16 bit di dati.
AUDxDR	Richiesta ad Agnus di una word di dati.
AUDxDSR	Richiesta ad Agnus di reinizializzare i puntatori ai dati.
dmasen	Abilitazione della richiesta di reinizializzazione.
percntrld	Ricarica il contatore del periodo, scritto dal 68000 tramite AUDxPER o scritto dal modo di collegamento.
percount	Decrementa di 1 il contatore del periodo.
perfin	Periodo giunto al termine del conteggio (valore = 1).
lencntrld	Ricarica il valore della lunghezza.
lencount	Decrementa di 1 il contatore della lunghezza.
lenfin	Lunghezza giunta al termine del conteggio (valore = 1).
volcntrld	Ricarica il valore del volume.
pbufld1	Ricarica il buffer di output tramite il valore di AUDxDAT.
pbufld2	Come pbufld1 ma solo per le transizioni 010→011 nel modo di collegamento.
AUDxAV	Collegamento del volume. Manda gli altri dati al registro di volume del canale successivo, anziché al convertitore D→A.
AUDxAP	Collegamento del periodo. Manda gli altri dati al registro del periodo del canale successivo, anziché al convertitore D→A.
penhi	Abilita il passaggio degli otto bit più significativi dei dati al convertitore D→A.
napnav	/AUDxAV * /AUDxAP + AUDxAV: nessun collegamento attivo, oppure solo il collegamento del volume. Condizione per il normale DMA e per le normali richieste di interrupt.
sq2,1,0	Il nome dei flip-flop di stato, dal più significativo al meno significativo.

Figura 5.8: Diagramma dello stato audio

6 HARDWARE DEL BLITTER

Introduzione

Il Blitter è uno dei due coprocessori dell'Amiga. Situato nel chip Agnus, viene utilizzato per copiare blocchi rettangolari di memoria da una regione a un'altra e per tracciare linee. Può trasferire circa 4 megabyte al secondo, il che equivale a una velocità circa doppia del 68000. Nel caso del tracciamento di linee la velocità è di quasi un milione di pixel al secondo.

Quando sposta blocchi di memoria, il Blitter può eseguire qualunque operazione logica su un massimo di tre aree sorgente, può far scorrere (shift) una o due di queste aree di un valore compreso fra 1 e 15 bit, può effettuare operazioni di riempimento di aree, e se necessario può mascherare la prima e l'ultima word di ogni riga.

Nel tracciamento può assegnare una diversa matrice a ogni linea oppure può disegnarle in modo tale che vi sia un solo pixel per linea orizzontale.

Il Blitter può accedere soltanto alla memoria chip. Tentare di utilizzarlo per leggere o scrivere memoria fast o altra memoria non chip, può avere come conseguenza la distruzione dei dati contenuti nella memoria chip.

Una singola operazione del Blitter, che sia il trasferimento di un blocco o il tracciamento di una linea, viene definita "blit" Ogni blit viene eseguito impostando gli opportuni registri, e l'ultima impostazione è quella del registro BLTSIZE, che dà il via alle operazioni. Dal momento che il Blitter è un coprocessore asincrono, il 68000 continua a funzionare anche mentre il blit è in esecuzione.

Caratteristiche della memoria

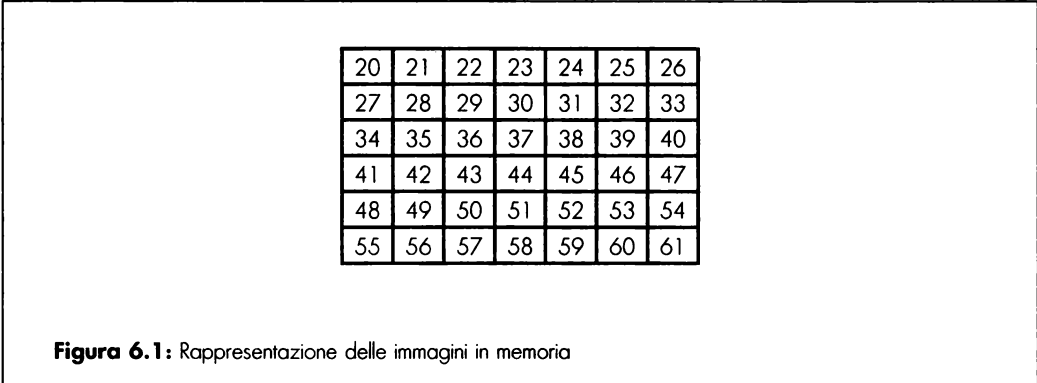
Si tenga presente che il Blitter opera su word, non su bit. Tutti i dati che legge, modifica e riscrive, sono costituiti da word di 16 bit. Tramite una corretta programmazione, comunque, il Blitter può eseguire molti tipi di operazioni che coinvolgano singoli bit.

Il Blitter è particolarmente adatto come supporto nelle operazioni grafiche. Per esempio, un

display 320 x 256 a 16 colori è composto da quattro bitplane di 10240 byte ciascuno. Ogni bitplane è formato da 256 righe di 40 byte, o meglio di 20 word.

I canali DMA

Il Blitter possiede quattro canali DMA: tre canali sorgente, denominati A, B e C, e un canale destinazione, chiamato D. Ogni canale è associato a una diversa serie di registri di puntamento, di modulo e di dati, e a un diverso bit di abilitazione. Due di essi possiedono registri di scorrimento, e uno possiede due registri per la mascheratura della prima e dell'ultima word di ogni riga. Tutti e quattro condividono lo stesso registro per le dimensioni del blit.

I registri di puntamento sono composti da due word, denominate BLTxPTH e BLTxPTL (in tutto il capitolo, la "x" nel nome sta a indicare uno dei quattro possibili canali: A, B, C o D.) Le due word risultano adiacenti all'interno dello spazio d'indirizzamento del 68000 che può pertanto inizializzarle con un unico trasferimento di 32 bit. Questi registri contengono valori espressi in byte, ma, poiché il Blitter opera solamente su word, il bit meno significativo viene ignorato. Dal momento che l'indirizzamento è limitato alla memoria chip, sono ignorati anche alcuni dei bit più significativi. Su macchine con 1 megabyte di memoria chip sono ignorati i 12 bit più significativi. Se fosse presente una maggior quantità di chip RAM sarebbero ignorati meno bit. In questi registri, comunque, è obbligatorio scrivere un indirizzo pari che corrisponda a RAM di tipo chip.

Ci si assicuri d'impostare a zero tutti i bit non utilizzati dai chip custom. Questi bit potrebbero essere usati in versioni future dell'Amiga. Inserire valori diversi da zero in questi bit potrebbe portare a risultati imprevisti.

Ognuno dei quattro canali DMA può essere abilitato o disabilitato indipendentemente dagli altri. I bit di abilitazione sono i bit SRCA, SRCB, SRCC, DEST contenuti nel registro di controllo BLTCON0.

Quando un canale viene disabilitato, non gli vengono più assegnati cicli di lettura in memoria; se si tratta di un canale sorgente, il valore costante contenuto nel corrispondente registro dati sarà utilizzato a ogni ciclo del Blitter. A questo scopo, i tre canali sorgente possiedono un registro dati impostabile tramite il 68000: BLTxDAT.

Le immagini in memoria si presentano di solito in maniera lineare: ogni word di dati si trova a un indirizzo immediatamente superiore a quello della word alla sua sinistra. Ogni linea è una continuazione della precedente.

La Figura 6.1 rappresenta un bitplane di un'immagine agli indirizzi 20-61. Ognuno di questi

20	21	22	23	24	25	26
27	28	29	30	31	32	33
34	35	36	37	38	39	40
41	42	43	44	45	46	47
48	49	50	51	52	53	54
55	56	57	58	59	60	61

Figura 6.1: Rappresentazione delle immagini in memoria

indirizzi contiene una word (16 pixel). Se quest'immagine fosse a 16 colori occorrerebbero quattro bitplane dello stesso tipo, e quindi quattro operazioni di copia per trasferirla completamente.

Il Blitter è molto efficiente in questo tipo di operazioni, dal momento che deve conoscere soltanto l'indirizzo iniziale (20), l'indirizzo destinazione e le dimensioni del blocco (altezza = 6, larghezza = 7). Non appena il bus dati diviene disponibile, il Blitter sposta automaticamente i dati, una word per volta. Il completamento del blit viene segnalato tramite un apposito flag e un interrupt.

Si noti che queste operazioni coinvolgono la memoria, e pertanto possono modificare l'immagine presente sul monitor.

Tutti i trasferimenti di dati compiuti dal Blitter si basano su regioni di memoria rettangolari. I canali DMA utilizzano il registro BLTSIZE per indicare altezza e larghezza delle righe. La larghezza può variare da 1 word a 64 (cioè da 16 a 1024 bit). L'altezza, invece, può variare da 1 linea a 1024. La larghezza viene scritta nei sei bit meno significativi del registro BLTSIZE. Se tutti questi bit sono a zero, viene usato il valore 64. Si noti che la larghezza è l'unico parametro del Blitter che venga espresso in word. L'altezza corrisponde invece ai 10 bit più significativi del registro BLTSIZE, con il valore 0 che rappresenta 1024 linee. Perciò, il più grande blit possibile è di 1024 x 1024 pixel. Vi sono però particolari operazioni di scorrimento e mascheratura che richiedono il trasferimento di una word extra di dati per ogni linea, riducendo l'effettivo limite orizzontale a 1008 pixel (1024 - 16).

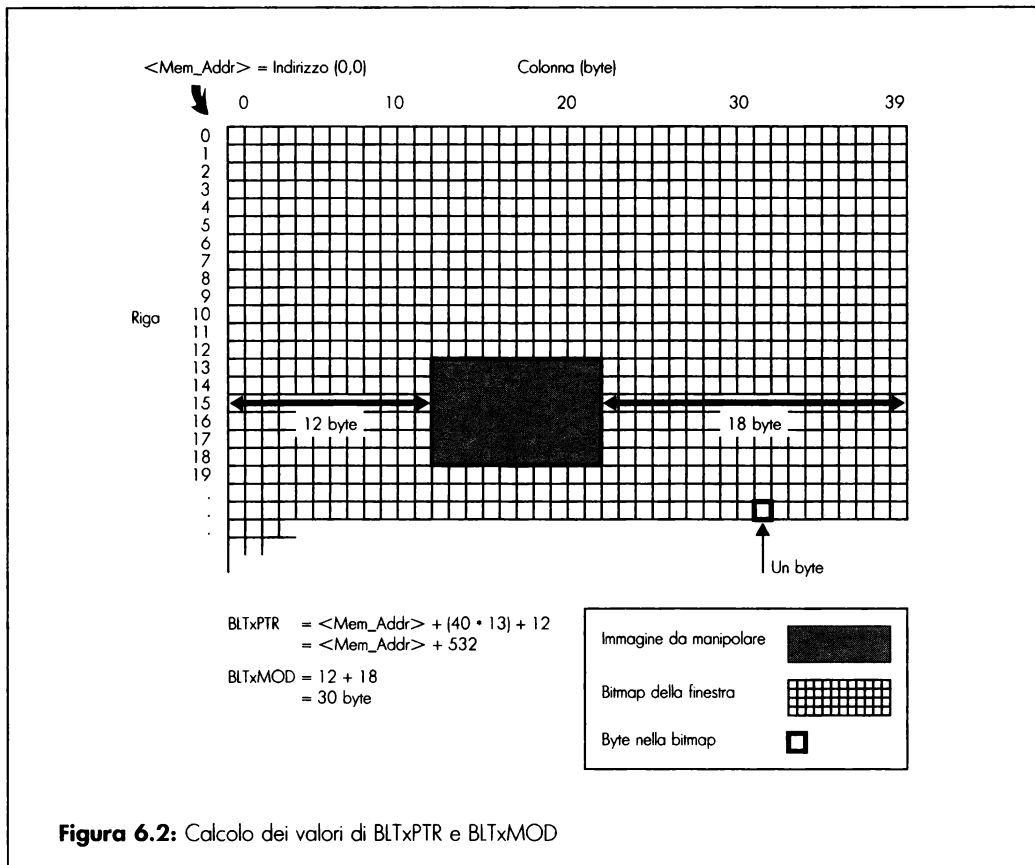



Figura 6.2: Calcolo dei valori di BLTxPTR e BLTxMOD

Riassumendo: la larghezza è espressa in word, e lo 0 rappresenta 64 word; l'altezza è espressa in linee, e lo 0 rappresenta 1024 linee.

Tramite i registri di modulo, il Blitter è anche in grado di accedere a immagini di dimensioni inferiori a quelle di un intero bitplane. Tutti i canali DMA hanno un registro di questo tipo, chiamato BLTxMOD. Ogni volta che viene trasferita una word di dati per un certo canale, il registro di puntamento viene incrementato di 2 (due byte = una word). Al termine di ogni riga, il modulo (un valore a 16 bit con segno) viene sommato al valore corrente del registro di puntamento (le dimensioni della riga vengono ricavate da BLTSIZE.)

Si noti che il valore del modulo è espresso in byte, non in word. Dal momento che il Blitter può operare soltanto sulle word, il bit meno significativo viene ignorato. Il segno del valore fornito viene esteso all'intera larghezza del registro di puntamento. In molti casi, infatti possono essere utili moduli negativi, come ad esempio per ripetere più volte la stessa riga impostando il modulo a un valore negativo uguale (in valore assoluto) alla larghezza della riga.

Per chiarire il concetto facciamo un esempio. Si supponga di voler operare su un bitplane completo di 320 x 256 pixel, e in particolare su un settore che inizia alla linea 13 e al byte 12 (il conteggio inizia dal valore zero) e che ha una larghezza di 10 byte e un'altezza di 6 byte. Per avere la corretta posizione iniziale occorre allora inizializzare il registro di puntamento all'indirizzo del bitplane, sommandovi 40 byte x 13 linee più 12 byte. La larghezza sarà di 5 word (10 byte) e l'altezza di 6 linee. Alla fine di ogni linea si dovranno saltare 30 byte per portarsi all'inizio della successiva, e questo sarà quindi il valore da inserire nel registro BLTxMOD. In genere, la larghezza della regione (espressa in word) moltiplicata per due e sommata al valore del modulo (espresso in byte) dovrebbe eguagliare la larghezza del bitplane contenente l'immagine.

Il Blitter può essere utilizzato anche per manipolare regioni lineari di dati: basta impostare a 1 l'altezza o la larghezza in BLTSIZE.

Dal momento che ogni canale DMA ha un registro di modulo personale, i dati possono essere trasferiti anche tra bitplane di dimensioni diverse. Ciò si rivela particolarmente utile per spostare piccole immagini in bitplane più grandi.

Il generatore di funzioni logiche

Il Blitter può combinare in 256 modi diversi i dati provenienti dai tre canali DMA sorgente, allo scopo di generare i dati da memorizzare tramite il canale destinazione. Le sorgenti possono essere bitplane di tre diverse immagini grafiche. Ogni pixel appartenente a queste regioni rettangolari sarà sottoposto alla medesima funzione logica, che pertanto andrà scelta in base alle otto possibili combinazioni dei relativi bit.

Per ogni combinazione occorre specificare se il corrispondente bit destinazione dovrà essere impostato o azzerato. Cerchiamo di "visualizzare" questo concetto ricorrendo a una tabella della verità come quella che segue. Vi sono riportati i tre canali sorgente e tutti i possibili valori delle combinazioni dei bit.

A	**B**	**C**	**D**	**Posizione in BLTCON0**	**Miniterm**
0	0	0	?	0	$\overline{ABC}$
0	0	1	?	1	$\overline{AB}C$
0	1	0	?	2	$\overline{A}B\overline{C}$

0	1	1	?	3	$\overline{A}BC$
1	0	0	?	4	$A\overline{BC}$
1	0	1	?	5	$A\overline{B}C$
1	1	0	?	6	$AB\overline{C}$
1	1	1	?	7	ABC

L'informazione viene memorizzata nel byte di controllo LF contenuto nel registro BLTCON0. Questo byte indica quale delle 256 operazioni logiche possibili va eseguita sulle tre sorgenti di un dato blit.

Per ricavare questo valore, si riempie la tabella della verità con i valori desiderati per D, e quindi si legge dal basso verso l'alto la colonna che ne risulta.

Per esempio, volendo impostare a 1 i bit destinazione dove risultano a 1 i bit della sorgente A oppure della sorgente B, occorrerebbe scrivere "1" nelle ultime quattro posizioni della colonna D (dove è a 1 il bit A), e nelle posizioni 2, 3, 6, 7 (dove è a 1 il bit B). Nelle rimanenti posizioni, la 0 e la 1, andrebbe invece scritto "0", poiché né A né B risultano impostati. Leggendo quindi la colonna D dal basso verso l'alto otteniamo 11111100, ovvero $FC.

Come secondo esempio, un byte di valore $80 (10000000 in binario) fa sì che nel canale destinazione vengano impostati a 1 solo i bit dove i corrispondenti bit delle sorgenti A, B e C sono tutti a 1 (ABC = 1, posizione 7 di LF). Tutti gli altri punti del rettangolo che corrispondono ad altre combinazioni di A, B e C vengono azzerati.

COME COSTRUIRE IL BYTE LF TRAMITE I MINTERM

Un metodo per ricavare il byte LF è costituito dalle equazioni logiche. Ognuna delle righe della tabella di verità corrisponde a un diverso "minterm", ovvero a una diversa assegnazione dei valori dei bit A, B e C. Per esempio, il primo minterm viene usualmente indicato con $\overline{ABC}$, cioè "not A and not B and not C". L'ultimo viene indicato con ABC.

Nota: porre due termini uno di seguito all'altro indica un'operazione di "and" logico, mentre invece un segno "+" indica un'operazione di "or" logico. L'operazione di "and" ha la precedenza su quella di "or", per cui AB + BC equivale a (AB) + (BC).

Qualunque funzione logica può essere espressa tramite somma di minterm. Supponiamo di voler calcolare la funzione che imposta a 1 i bit di D quando A = 1 e C = 0 oppure quando B = 1. Quest'operazione può essere scritta $A\overline{C} + B$, ovvero "A and not C or B". A questo punto, dal momento che "1 and A" corrisponde ad A:

$$D = A\overline{C} + B$$

$$D = A(1)\overline{C} + (1)B(1)$$

Dal momento che A or $\overline{A}$ è sempre vera ($A + \overline{A} = 1$) e ciò vale anche per B e per C, possiamo sviluppare ulteriormente l'equazione:

$$D = A(1)\overline{C} + (1)B(1)$$

$$D = A(B + \overline{B})\overline{C} + (A + \overline{A})B(C + \overline{C})$$

$$D = AB\overline{C} + A\overline{B}\overline{C} + AB(C + \overline{C}) + \overline{A}B(C + \overline{C})$$

$$D = AB\overline{C} + A\overline{B}\overline{C} + ABC + AB\overline{C} + \overline{A}BC + \overline{A}B\overline{C}$$

Eliminando i doppioni, rimangono cinque minterm:

$$D = A\overline{C} + B = AB\overline{C} + A\overline{B}\overline{C} + ABC + \overline{A}BC + \overline{A}B\overline{C}$$

che corrispondono rispettivamente alle posizioni 6, 4, 7, 3 e 2 di BLTCON0. Queste sono le posizioni che andranno impostate a 1 nella tabella.

L'ampia gamma di operazioni logiche possibili permette l'uso di sofisticate tecniche grafiche. Per esempio, con pochi blit è possibile muovere l'immagine di un'automobile di fronte ad alcuni edifici disegnati in precedenza. Questo effetto richiede la presenza in memoria delle immagini dell'automobile e degli edifici (lo sfondo), e di una maschera dell'automobile che abbia impostati a 1 tutti i bit corrispondenti ai pixel non trasparenti dell'immagine. Questa maschera può essere considerata "l'ombra" dell'automobile che sarebbe prodotta da una sorgente di luce coincidente con l'osservatore.

La maschera richiede un solo bitplane, indipendentemente dalla profondità dello sfondo, dal momento che è uguale per ogni bitplane.

Per produrre l'animazione dell'automobile, si salva innanzi tutto l'area di sfondo dove verrà collocata l'immagine. Quindi, con un secondo blit, vi si trasferisce l'immagine dell'automobile: il disegno è pronto per essere visualizzato. Per passare all'immagine successiva, è sufficiente ripristinare lo sfondo originale, calcolare la nuova posizione dell'automobile e ripetere i passi precedenti (questa tecnica dà risultati migliori se si utilizzano blit sincronizzati con la posizione del pennello elettronico o se si ricorre al double buffering.)

Per salvare lo sfondo si copia l'area rettangolare coinvolta (tramite il canale A, per esempio) in un buffer temporaneo (utilizzando il canale D). In questo caso la funzione logica da usare è "A", la funzione di semplice copia. Dalla Tavola 6.1 vediamo che essa corrisponde a un valore di LF pari a $F0.

Per disegnare l'automobile si utilizza il canale A per la maschera, il canale B per l'immagine vera e propria, il canale C per lo sfondo e il canale D per memorizzare il risultato.

È necessario leggere anche lo sfondo di destinazione, perché parte di esso potrebbe essere presente anche nell'immagine risultante.

Per questo blit dobbiamo usare una funzione logica che, dove la maschera in A è impostata a 1, lasci passare l'immagine proveniente da B, e che, dove la maschera è impostata a zero, lasci passare lo sfondo originale proveniente da C. Questa funzione, comunemente denominata di "cookie-cut", taglio del biscotto, corrisponde a un valore di LF pari a $CA, ovvero $AB + \overline{A}C$.

Per ripristinare lo sfondo originale, si ricorre nuovamente alla funzione di copia standard ($F0).

Modificando in maniera opportuna i registri di scorrimento e di mascheratura, e ripetendo ogni volta i tre passi ora descritti, l'automobile sembrerà muoversi attraverso lo sfondo.

Anche se non sempre si tratta del più efficace metodo di animazione, capita comunque molto spesso d'incontrare la funzione di "cookie-cut".

La Tavola 6.1 riporta alcune delle funzioni logiche più utilizzate e il loro valore.

Tavola 6.1: Valori dei minterm più comuni

Operazione logica	LF	Operazione logica	LF
$D = A$	$F0	$D = AB$	$C0
$D = \overline{A}$	$0F	$D = A\overline{B}$	$30
$D = B$	$CC	$D = \overline{A}B$	$0C
$D = \overline{B}$	$33	$D = \overline{AB}$	$03
$D = C$	$AA	$D = BC$	$88
$D = \overline{C}$	$55	$D = B\overline{C}$	$44
$D = AC$	$A0	$D = \overline{B}C$	$22
$D = A\overline{C}$	$50	$D = \overline{BC}$	$11
$D = \overline{A}C$	$0A	$D = A + \overline{B}$	$F3
$D = \overline{AC}$	$05	$D = \overline{A} + \overline{B}$	$3F
$D = A + B$	$FC	$D = A + \overline{C}$	$F5
$D = \overline{A} + B$	$CF	$D = \overline{A} + \overline{C}$	$5F
$D = A + C$	$FA	$D = B + \overline{C}$	$DD
$D = \overline{A} + C$	$AF	$D = \overline{B} + \overline{C}$	$77
$D = B + C$	$EE	$D = AB + \overline{A}C$	$CA
$D = \overline{B} + C$	$BB		

COME COSTRUIRE IL BYTE LF TRAMITE I DIAGRAMMI DI VENN

Un altro metodo per ricavare la funzione logica richiesta è l'uso dei diagrammi di Venn:

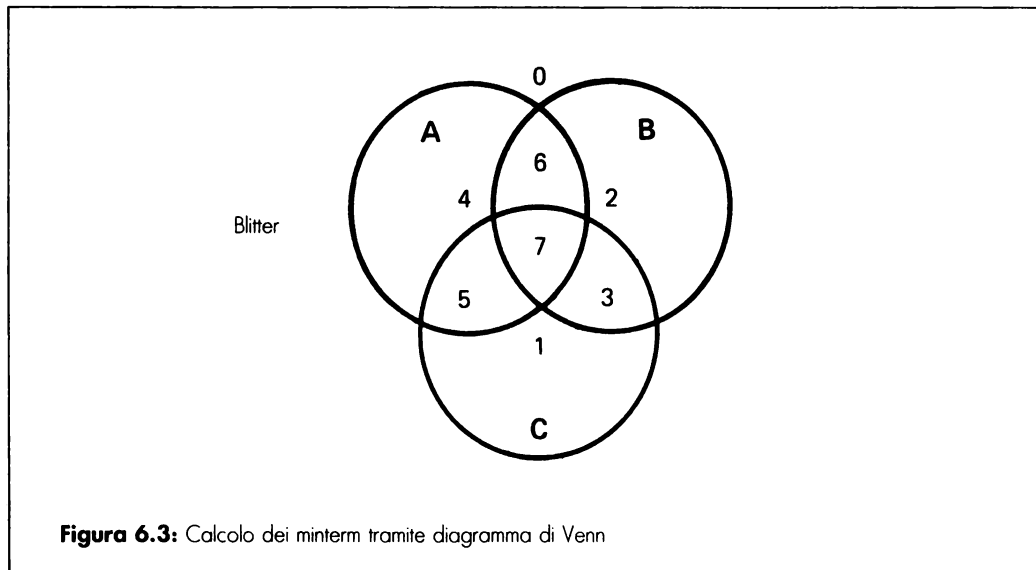



Figura 6.3: Calcolo dei minterm tramite diagramma di Venn

1. Per selezionare la funzione D = A, si scelgono i minterm completamente racchiusi dal cerchio A, ovvero i minterm 7, 6, 5 e 4, che danno origine al valore:

Numero del minterm	7 6 5 4 3 2 1 0
Minterm selezionati	1 1 1 1 0 0 0 0
	F 0 = $FO

2. Per selezionare una funzione che rappresenti la combinazione di due sorgenti, si scelgono tutti i minterm racchiusi da entrambi i cerchi, cioè quelli appartenenti alla loro intersezione. Per esempio, la combinazione AB è rappresentata dall'area comune ai cerchi A e B: i minterm 7 e 6.

Numero del minterm	7 6 5 4 3 2 1 0
Minterm selezionati	1 1 0 0 0 0 0 0
	C 0 = $CO

3. Per utilizzare una funzione che sia l'inverso, "not", di una sorgente, come $\overline{A}$, si scelgono tutti i minterm non racchiusi dal cerchio corrispondente. In questo caso, 0, 1, 2 e 3.

Numero del minterm	7 6 5 4 3 2 1 0
Minterm selezionati	0 0 0 0 1 1 1 1
	0 F = $0F

4. I minterm possono essere combinati tramite operazioni di "or". Per esempio, l'equazione AB + BC diventa:

Numero del minterm	7 6 5 4 3 2 1 0
AB	1 1 0 0 0 0 0 0
BC	1 0 0 0 1 0 0 0
AB + BC	1 1 0 0 1 0 0 0
	C 8 = $C8

Scorrimento (shift) e mascheratura

Finora abbiamo visto come si utilizza il Blitter per trasferire word di memoria e per combinarle tramite operazioni logiche. Ciò è sufficiente per muovere immagini grafiche i cui pixel non debbano mutare posizione rispetto all'inizio di una word. Se l'immagine della nostra automobile inizia al secondo pixel da sinistra, è particolarmente semplice trasferirla in una posizione in cui la sua immagine inizia ancora a due pixel dall'inizio di una word. Spesso, però, sarà necessario spostare l'automobile in posizioni diverse. A questo scopo, i canali A e B possiedono un circuito "barrel shifter" che può far scorrere un'immagine di una quantità variabile da 0 a 15 pixel.

Quest'operazione di shift è completamente "gratuita": per eseguire un blit con shift occorre cioè lo stesso tempo necessario per un blit senza shift, a differenza di quanto accadrebbe con il 68000. Lo scorrimento avviene in genere verso destra. Ciò permette di spostare le immagini un pixel alla volta, anche se l'indirizzamento viene eseguito a gruppi di 16 bit.

Ma se i dati scorrono verso destra, che cosa viene inserito a sinistra? Per la prima word del blit degli zeri, mentre per ogni word successiva vengono inseriti i dati provenienti dalla word precedente.

Il valore dello shift per il canale A viene impostato tramite i bit 15-12 di BLTCON0; quello relativo al canale B tramite i bit 15-12 di BLTCON1. Nella maggior parte dei casi il valore è lo stesso per entrambi i canali. Per scorrimenti maggiori di 15 bit si deve incrementare il valore contenuto nel registro di puntamento del canale destinazione: per uno shift di 100 bit occorre far avanzare il puntatore di 6 word (100/16) e quindi effettuare uno shift verso destra per i rimanenti 4 bit.

Si consideri per esempio un blit largo tre word e alto due, con uno shift di 4 bit. Per semplicità, s'immagini di fare una copia diretta da A a D. La prima word scritta in D sarà la prima word letta da A, dopo uno scorrimento verso destra di quattro bit, con altrettanti zeri inseriti a sinistra. La seconda word sarà uguale alla word successiva di A, spostata verso destra, con i quattro bit meno significativi della prima word inseriti a sinistra. Ciò si ripete fino all'ultima word del blit.

Nei blit in cui viene eseguito uno shift, si ha dunque un inserimento di zeri *solo nella prima word della prima riga*. Nelle prime word di tutte le altre righe sono contenuti invece i dati provenienti dalle ultime word delle righe precedenti. Per la maggior parte delle applicazioni grafiche ciò è inaccettabile. Ecco perché il Blitter ha la capacità di mascherare la prima e l'ultima word di ogni riga, purché provengano dal canale A. In questo modo diventa possibile trasferire una regione rettangolare di dati i cui estremi non coincidono esattamente con una word. I due

registri coinvolti si chiamano BLTAFWM e BLTALWM. Se non vengono utilizzati, devono essere inizializzati con il valore $FFFF.

Le fonti-carattere dell'Amiga sono memorizzate in bitmap in formato "compresso" (cioè un carattere di seguito all'altro, senza spazi). I caratteri vengono poi estratti dal Blitter, mascherando i bit indesiderati, e poi vengono collocati in qualunque posizione orizzontale tramite lo shift appropriato.

Prima di qualunque scorrimento le maschere subiscono un'operazione di and logico con i dati sorgente. Solo quando il corrispondente bit della maschera è a 1, il bit di A appare nell'operazione logica. La prima word di ogni riga viene mascherata tramite BLTAFWM, l'ultima tramite BLTALWM. Se la larghezza di una riga è di una sola word, entrambe le maschere vengono applicate simultaneamente.

Queste maschere sono anche utili per estrarre un certo intervallo di colonne da un bitplane. Si immagini, per esempio, di avere un rettangolo predisegnato contenente un'immagine larga 23 pixel. Il bordo sinistro corrisponde al bordo sinistro della bitmap, che è larga due word. Si supponga di voler trasferire questo rettangolo nella posizione 5 (il quinto pixel) del nostro schermo 320 x 256, senza disturbare ciò che giace all'esterno del rettangolo.

Per far ciò, si punta il canale B alla bitmap contenente l'immagine sorgente, e i canali C e D alla bitmap dello schermo. Si utilizza un valore di shift pari a 5 e una larghezza di blit pari a 2 word. Dev'essere eseguita una semplice operazione di copia, ma devono essere lasciati indisturbati i primi 5 bit della prima word e gli ultimi quattro dell'ultima (2 * 16 – (23 + 5)).

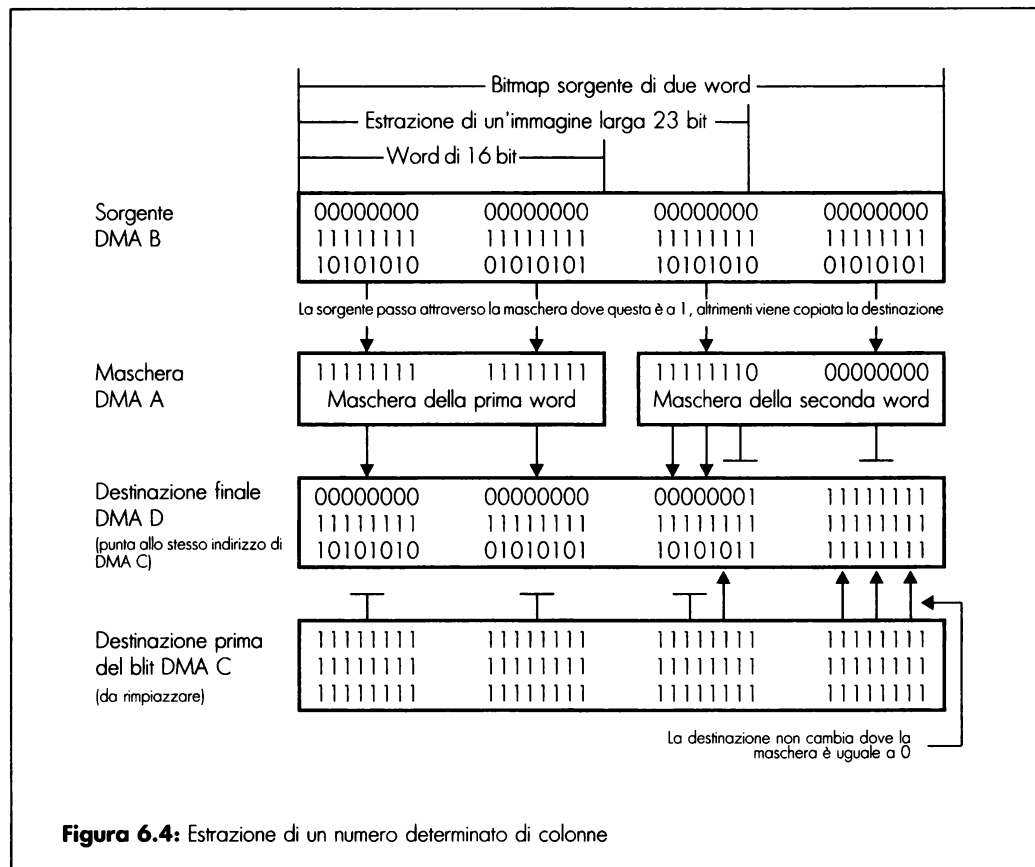



Figura 6.4: Estrazione di un numero determinato di colonne

Sfruttiamo quindi il canale A: si inizializza il registro BLTADAT a $FFFF; si imposta inoltre il registro BLTAFWM con i 5 bit più significativi a zero ($07FF) e BLTALWM con i quattro bit meno significativi a zero ($FFF0). Non si abilita il canale A, ma solo i canali B, C e D, dal momento che A dev'essere utilizzato come una semplice maschera, uguale per ogni word. I dati in B (sorgente) devono passare alla destinazione D solo quando il canale A è a 1 (minterm AB), e devono passare invece i dati originali della destinazione (canale C) quando A è a 0 (minterm $\overline{A}$C), il che dà origine alla classica funzione di "cookie-cut", ovvero $CA.

Nota: anche se il canale A non è abilitato, viene utilizzato ugualmente nella funzione logica, impostando preventivamente il registro dati. Disabilitare un canale disattiva semplicemente il trasferimento dei dati dalla memoria: tutte le altre operazioni vengono eseguite ugualmente, sulla base della costante contenuta nel registro dati.

Un metodo alternativo, ma più sottile, per ottenere il medesimo risultato è quello di usare per A un valore di shift pari a 5, la maschera della prima word tutta a 1 e la maschera dell'ultima word con i nove bit meno significativi azzerati. Tutti gli altri registri rimangono invariati.

Attenzione: i registri dati devono essere impostati soltanto *dopo* aver impostato i corretti valori di shift. Fare il contrario può condurre a risultati indesiderati. Per esempio, se dopo l'ultima utilizzazione del Blitter il valore di BSHIFT nel registro BLTCON1 è 4, dopo aver caricato in BLTBDAT il valore $FF e in BSHIFT 2 (in quest'ordine), il valore di BLTBDAT che sarà utilizzato sarà $FF >> 4 e non $FF >> 2. L'atto d'impostare un registro, infatti, fa sì che il dato sia immediatamente sottoposto all'ultima operazione di shift impostata.

Modo discendente

Il metodo di copia visto finora funziona perfettamente se l'area sorgente e quella destinazione non si sovrappongono. Volendo muovere un'immagine di una riga verso il basso (indirizzi crescenti), ci si trova di fronte a un nuovo problema: la seconda riga viene modificata prima che possa essere letta! Il Blitter possiede un particolare modo operativo, il modo discendente, che risolve elegantemente questo problema.

Il modo discendente viene attivato tramite l'impostazione del bit 1, BLITREVERSE di BLTCON1. Utilizzando questo modo, per ogni word trasferita i puntatori vengono *decrementati* di due anziché incrementati, e anche il modulo viene sottratto anziché sommato. Lo scorrimento avviene verso sinistra, BLTFWM si riferisce all'ultima word della riga (che è comunque la prima a essere letta) e BLTLWM alla prima.

Perciò, per una normale operazione di copia, la sola differenza nell'impostazione del Blitter (supponendo che non vi siano shift e mascherature) consiste nell'inizializzare i registri di puntamento all'ultima word del blocco piuttosto che alla prima. Il valore del modulo, le dimensioni e gli altri parametri rimangono gli stessi.

Si tratta di una tecnica diversa da quella dell'indirizzamento con predecremento, disponibile sul 68000, dove il registro d'indirizzamento dev'essere impostato in modo da puntare non all'ultima word del blocco, ma a quella immediatamente successiva.

Il modo discendente viene utilizzato anche per il riempimento di aree tramite il Blitter, che discuteremo in seguito.

Copia di regioni arbitrarie

Uno degli usi più comuni del Blitter consiste nello spostare blocchi rettangolari di dati da un bitplane a un altro, o in una diversa posizione all'interno dello stesso bitplane. Questi blocchi,

in genere, non coincidono esattamente con i confini delle word da cui sono costituiti i bitplane, per cui si rendono necessari scorrimenti e mascherature. Si possono poi presentare numerose altre complicazioni, ed è probabile che siano necessari un certo tempo e una certa quantità di tentativi ed errori prima di raggiungere una comprensione completa degli argomenti discussi in questo paragrafo.

Un'immagine sorgente che occupi solo due word può richiedere il trasferimento di tre word, se viene copiata con certi valori di shift. Per esempio, se facciamo scorrere di 12 bit il rettangolo di 23 pixel dell'esempio precedente, è necessaria una larghezza di tre word. Al contrario, un'immagine contenuta in tre word distinte può richiedere solo due word per il trasferimento, se viene copiata con particolari valori di shift. In tutti questi casi, la larghezza del blit dev'essere uguale al maggiore dei due valori, in modo da comprendere sia la sorgente che la destinazione. Poi, per eliminare i dati non desiderati, dev'essere applicata un'opportuna mascheratura.

Ecco alcune indicazioni generali da utilizzare nella copia di regioni.

1. Utilizzare il canale A, disabilitato, preinizializzato a $FFFF, insieme alla maschera e agli appropriati valori di shift per mascherare la funzione di cookie-cut. Utilizzare il canale B per i dati sorgente, il canale C per i dati provenienti dalla destinazione e il canale D per scrivere i dati risultanti. La funzione da utilizzare è $CA.

2. Se si esegue uno shift, utilizzare il modo ascendente se lo shift è verso destra, il modo discendente se lo shift è verso sinistra (questi shift rappresentano lo spostamento del bordo dell'immagine all'interno di una word e non sono shift assoluti, come è stato spiegato in precedenza).

3. Se sorgente e destinazione si sovrappongono, utilizzare il modo ascendente se la destinazione ha un indirizzo minore della sorgente (è più in alto sullo schermo), e il modo discendente in caso contrario.

4. Se la sorgente occupa un numero di word di larghezza superiore a quello della destinazione, si utilizzi lo stesso valore di shift per i canali A e B e si impostino BLTAFWM e BLTALWM per mascherare opportunamente i dati sorgente in B.

5. Se la destinazione occupa un numero di word di larghezza superiore a quello della sorgente, si utilizzi un valore di shift pari a 0 per il canale A e si impostino BLTAFWM e BLTALWM per mascherare opportunamente i dati destinazione in D.

6. Se sorgente e destinazione hanno la stessa larghezza in word, si utilizzi il canale A per mascherare l'una o l'altra.

Le condizioni 2 e 3 possono essere contraddittorie se, per esempio, si vuole muovere un'immagine di un pixel in basso e a destra. In tal caso, infatti, occorre utilizzare il modo discendente in modo che la destinazione non modifichi la sorgente prima che quest'ultima venga letta, ma d'altra parte uno shift verso destra richiede il modo ascendente. In alcuni casi è possibile aggirare la condizione 2 con un'appropriata mascheratura, ma può accadere che mascherare la prima e l'ultima word non sia sufficiente. È possibile allora costruire in memoria la maschera corrispondente a una singola linea di dati e abilitare il canale A perché ne faccia uso. Impostando il modulo di A al valore della lunghezza di una riga, cambiato di segno, la maschera viene riutilizzata per ogni riga.

Riempimento di aree tramite il Blitter

In aggiunta al trasferimento di dati, il Blitter può simultaneamente eseguire un'operazione di riempimento durante la copia. Quest'operazione ha una sola restrizione: l'area in questione dev'essere delimitata da linee che presentino un solo pixel per ogni riga orizzontale. Proprio a questo scopo è disponibile un modo speciale per il tracciamento di linee. Innanzitutto si deve utilizzare una normale funzione di copia (o qualunque altra funzione, dal momento che il riempimento ha luogo dopo tutte le altre operazioni), e si deve ricorrere al modo discendente. Ora s'imposti il flag di riempimento inclusivo (FILL_OR, bit 3 di BLTCON1) oppure il flag di riempimento esclusivo (FILL_XOR, bit 4 di BLTCON1). Il modo inclusivo riempie le aree comprese tra due linee, lasciando intatte queste ultime. Il modo esclusivo fa lo stesso, ma, pur lasciando inalterata la linea di delimitazione di destra, cancella quella di sinistra. Ciò dà origine ad aree più strette di un pixel rispetto a quelle generate tramite il modo inclusivo.

Per esempio, il pattern:

    00100100-00011000

tramite il riempimento inclusivo diventa:

    00111100-00011000

usando il modo esclusivo si otterrebbe invece:

    00011100-00001000

Vi è un altro bit (FILL_CARRYIN o FCI, il bit 2 di BLTCON1) che causa il riempimento dell'area esterna alle linee di delimitazione. L'esempio precedente, in modo inclusivo avrebbe dato origine a:

    11100111-11111111

mentre, in modo esclusivo:

    11100011-11110111

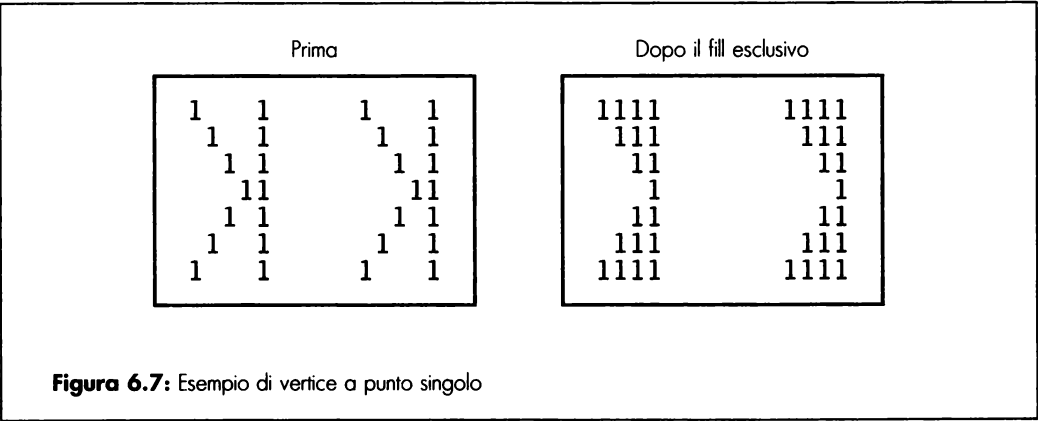
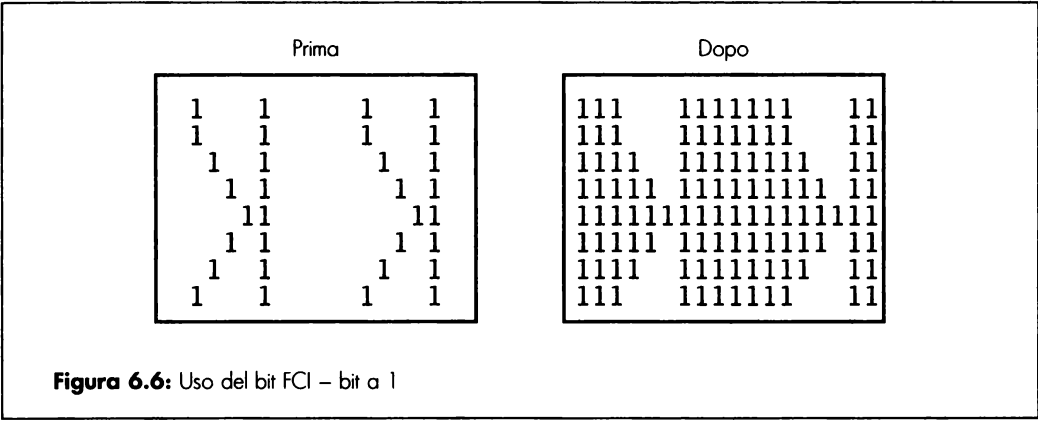
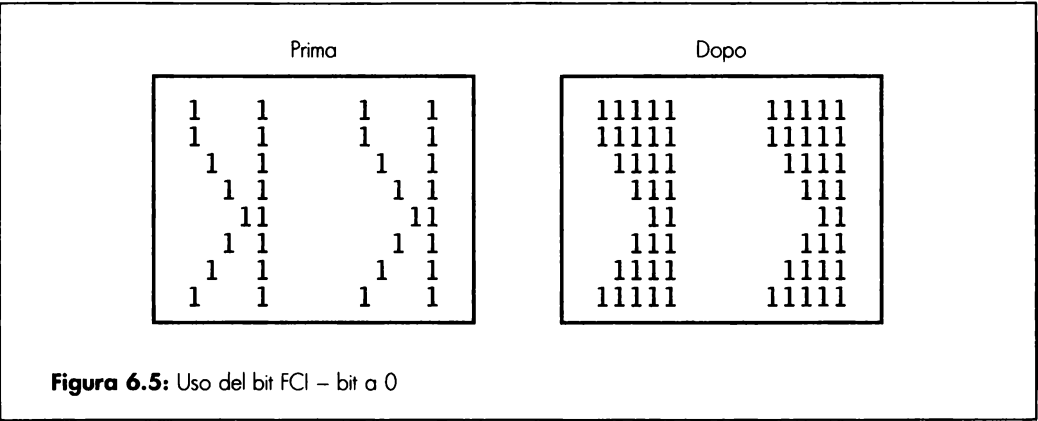
Se il bit FCI è a 1, l'area all'esterno delle linee viene riempita con "1" mentre l'area al loro interno viene lasciata a "0".

Volendo produrre vertici appuntiti costituiti da un solo pixel, occorre utilizzare il modo esclusivo.

Il Blitter utilizza il bit FCI come stato iniziale nel corso del riempimento, a partire dall'estremità destra di ogni linea. Per ogni bit a 1 incontrato nell'area sorgente, questo stato viene invertito, causando o meno il riempimento con degli 1 degli eventuali 0 successivi. Ciò continua per tutta la lunghezza della linea fino all'estremità sinistra.

Figura 6.5: Uso del bit FCI – bit a 0

Figura 6.6: Uso del bit FCI – bit a 1

Figura 6.7: Esempio di vertice a punto singolo

Flag di "Blitter done"

Il blit ha inizio quando viene impostato il registro BLTSIZE. Il 68000 non si ferma mentre il Blitter è al lavoro: i due processori possono funzionare contemporaneamente, e ciò spiega in parte la velocità che caratterizza l'Amiga. È necessario però prestare una certa attenzione nell'uso del Blitter.

Nel registro DMACONR è presente un flag di blit completato, detto anche flag di "Blitter done", o DMAF_BLTDONE. Questo flag risulta impostato quando è in corso un blit.

Se un blit è stato iniziato ma per qualche motivo non ha ancora potuto accedere alla memoria (per esempio a causa del DMA dei playfield) il bit potrebbe non essere ancora impostato. Il 68000, dal canto suo, potrebbe, nello stesso momento, utilizzare memoria fast o il suo cache interno, e continuare quindi ad avere cicli di memoria.

La soluzione a questo problema consiste nel leggere un indirizzo in memoria chip o un registro hardware prima di controllare il flag. Un esempio potrebbe essere:

```
btst.b  #DMAB_BLTDONE-8,DMACONR(a1)
btst.b  #DMAB_BLTDONE-8,DMACONR(a1)
```

dove a1 contiene l'indirizzo base dei chip custom. La prima lettura potrebbe non fornire un risultato corretto, ma la seconda lo fornisce di sicuro.

Con il chip Fat Agnus, il Blitter imposta il flag subito dopo l'accesso a BLTSIZE (che serve per dare il via al blit), anziché dopo aver ottenuto il primo ciclo di DMA. Per i modelli che non montano questo chip conviene adottare il metodo descritto.

IL BLITTER E IL MULTITASKING

Quando è in corso un blit, non si deve accedere a nessun registro del Blitter. Per maggiori dettagli sull'uso del Blitter in ambiente multitasking si consultino gli altri manuali di questa serie, e in particolare le descrizioni delle funzioni OwnBlitter( ) e DisownBlitter( ). Anche quando si ottiene dal sistema l'uso esclusivo del Blitter, questo potrebbe ancora essere impegnato a portare a termine un blit precedente, per cui è necessario controllare il flag di Blitter done. Si raccomanda l'uso della funzione WaitBlit( ).

Il flag di Blitter done va controllato anche prima di riutilizzare il risultato di un'operazione di blit. Il blit potrebbe non essere ancora terminato e i dati potrebbero non essere ancora pronti. Ciò rende difficile localizzare certi particolari bug; per esempio, un 68000 potrebbe essere sufficientemente lento da permettere il completamento di un blit senza aver bisogno di controllare questo flag, mentre un 68020, sfruttando il suo cache buffer di istruzioni, potrebbe accedere ai dati prima che il Blitter avesse tempo di terminarne la scrittura.

Si immagini di avere una sotto-routine che visualizzi temporaneamente un riquadro di testo sul monitor, sovrapponendolo alle immagini preesistenti. Questa sotto-routine potrebbe allocare un'area di memoria per ospitare i dati originali durante la visualizzazione del testo, quindi procedere alla visualizzazione vera e propria. Al termine, la sotto-routine potrebbe utilizzare il Blitter per ripristinare l'immagine originale e liberare la memoria allocata. Se la memoria viene liberata prima di controllare il flag di blit completato, qualche altro processo potrebbe allocarla e riutilizzarla prima della fine del blit, distruggendo perciò i dati dell'immagine.

Flag di interrupt

Il Blitter possiede inoltre un flag di interrupt, che viene impostato al termine di ogni blit. Questo flag (INTF_BLIT) genera un interrupt del 68000. Per maggiori informazioni sugli interrupt si consulti il Capitolo 7.

Flag di zero

Esiste inoltre un flag di zero, che può essere controllato per sapere se l'operazione logica appena conclusa ha dato un risultato uguale a 0 per tutti i bit coinvolti (anche nel caso che la destinazione non sia stata realmente scritta, perché il canale D era disabilitato). Questa caratteristica è utile nella rilevazione delle collisioni: per sapere se vi è sovrapposizione basta ricorrere a un'operazione di and logico su due aree sorgenti. Se le immagini non si sovrappongono, il flag risulta impostato.

Il flag ha significato soltanto quando l'operazione di blit è stata completata, e corrisponde al bit DMAF_BLTNZERO del registro DMACONR.

Pipeline dei registri

Il Blitter esegue molte operazioni in un solo ciclo: scorrimenti, mascherature, operazioni logiche, riempimenti di aree e controlli di azzeramenti. Per far sì che un così alto numero di operazioni avvenga nel più breve tempo possibile, il Blitter è dotato di un sistema interno di pipeline. Ciò significa che, invece di compiere tutte le operazioni in un unico ciclo, esso suddivide ogni operazione in due cicli successivi (per semplicità, usiamo la parola "ciclo" in maniera piuttosto libera). Si può immaginare il Blitter come se fosse composto di due chip collegati in serie. A ogni ciclo, il primo chip compie le sue operazioni sui nuovi dati in ingresso. Li passa quindi, parzialmente elaborati, al secondo chip, che termina le operazioni nel secondo ciclo, quando il primo chip è già occupato da una nuova serie di dati in ingresso. Ogni serie di dati richiede due cicli completi per passare attraverso ai due chip.

Ciò significa che in realtà vengono lette due serie complete di dati sorgente prima che i primi dati vengano scritti nella destinazione. Questo consente, per esempio, di far scorrere di una word verso destra una bitmap usando il modo ascendente, anche se, teoricamente, parte della sorgente dovrebbe essere riscritta prima ancora di essere letta.

Tavola 6.2: Sequenza tipica di un ciclo del Blitter

USO in BLTCON0	**Canali attivi**	**Sequenza dei cicli**
F	A B C D	A0 B0 C0 - A1 B1 C1 D0 A2 B2 C2 D1 D2
E	A B C	A0 B0 C0 A1 B1 C1 A2 B2 C2
D	A B D	A0 B0 - A1 B1 D0 A2 B2 D1 - D2
C	A B	A0 B0 - A1 B1 - A2 B2
B	A C D	A0 C0 - A1 C1 D0 A2 C2 D1 - D2
A	A C	A0 C0 A1 C1 A2 C2

9	A D	A0 - A1 D0 A2 D1 - D2
8	A	A0 - A1 - A2
7	B C D	B0 C0 - - B1 C1 D0 - B2 C2 D1 - D2
6	B C	B0 C0 - B1 C1 - B2 C2
5	B D	B0 - - B1 D0 - B2 D1 - D2
4	B	B0 - - B1 - - B2
3	C D	C0 - - C1 D0 - C2 D1 - D2
2	C	C0 - C1 - C2
1	D	D0 - D1 - D2
0	nessuno	- - - -

Note riguardanti la tavola precedente:

- Riempimento aree disattivato.
- Nessuna attività concorrente sul bus.
- Blit di tre word.
- Una tipica operazione del Blitter richiede una doppia lettura di tutte le sorgenti prima che siano disponibili i primi dati destinazione (si veda la precedente discussione). Particolare cura va adottata per regioni che si sovrappongano.

Questa tavola vuol essere soltanto un esempio del tipico ordine in cui vengono eseguiti i cicli di lettura/scrittura del Blitter sul bus dati. Questi cicli sono allocati dinamicamente a seconda del modo operativo del Blitter e delle attività concorrenti sul bus: 68000, playfield e altri canali DMA. La Commodore-Amiga non garantisce che questa tavola verrà rispettata dai modelli futuri.

Tracciamento di linee tramite il Blitter

Oltre a tutte le funzioni viste finora, il Blitter è in grado di tracciare linee con una determinata matrice (pattern). Questo modo operativo viene attivato mediante l'impostazione del bit 0 (LINEMODE) di BLTCON1. Le linee possono raggiungere una lunghezza massima di 1024 pixel, possono essere tracciate in vari modi, avere una determinata matrice o avere un solo pixel per ogni linea orizzontale, come viene richiesto dallo stesso Blitter nel tracciamento di aree.

Molti registri del Blitter assumono un significato diverso durante il tracciamento di linee. Si consulti l'Appendice A per una descrizione più dettagliata del significato dei registri e dei loro bit.

Per tracciare una linea, il Blitter congiunge fra loro i due estremi, quello iniziale e quello finale. L'operazione può essere rappresentata tramite un vettore. La direzione e il verso di questo vettore individuano uno degli otto ottanti rappresentati nella Figura 6.8. Nel grafico le x crescono verso destra e le y verso il basso. Il numero tra parentesi è il numero dell'ottante, l'altro numero rappresenta il valore da inserire nei bit 4-2 di BLTCON1.

Basarsi sugli ottanti consente di semplificare le operazioni del Blitter sfruttando la simmetria esistente tra x e $-x$ e tra y e $-y$. La tavola seguente elenca gli ottanti e i valori corrispondenti.

Figura 6.8: Ottanti per il tracciamento di linee

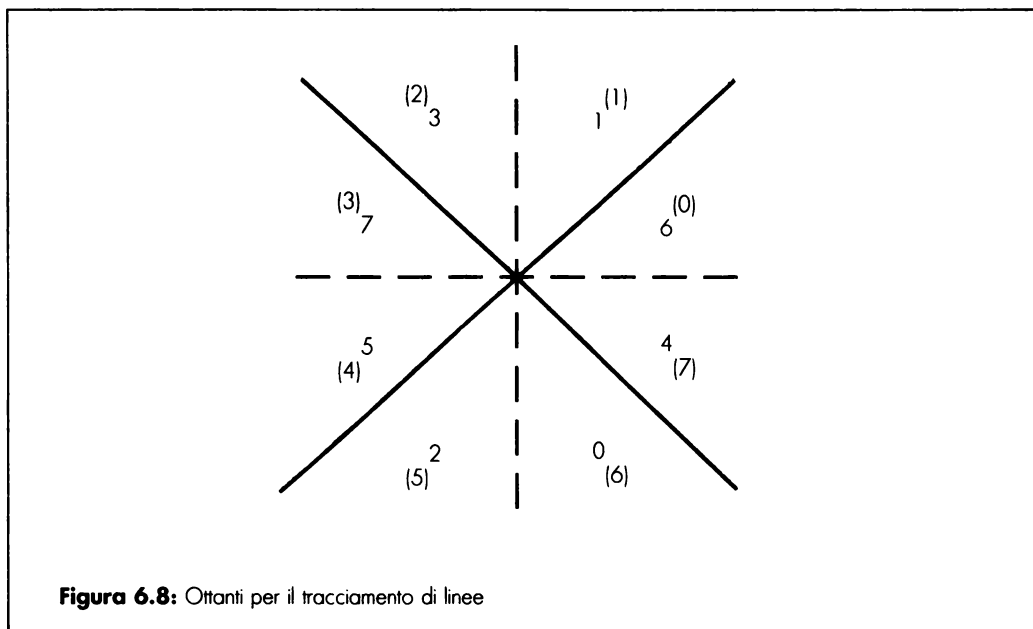



Tavola 6.3: Bit di BLTCON1 nel tracciamento di linee

Bit di BLTCON1 **4 3 2**	**Ottante**
1 1 0	0
0 0 1	1
0 1 1	2
1 1 1	3
1 0 1	4
0 1 0	5
0 0 0	6
1 0 0	7

I bit 4-2 di BLTCON1 vanno inizializzati seguendo la Tavola 6.3. Fatto ciò, occorre considerare le variabili *dx* e *dy*, intese come il valore assoluto della differenza rispettivamente delle coordinate *x* e delle coordinate *y*.

```
dx = abs(x2 - x1);
dy = abs(y2 - y1);
```

Ora è necessario riordinarle in modo che *dx* sia sempre maggiore di *dy*.

```
if (dx < dy)
    {
    temp = dx;
    dx = dy;
    dy = temp;
    }
```

In alternativa, *dx* e *dy* possono essere impostati in questo modo:

```
dx = max(abs(x2 - x1), abs(y2 - y1));
dy = min(abs(x2 - x1), abs(y2 - y1));
```

Questi calcoli hanno l'effetto di "normalizzare" la linea all'interno dell'ottante 0. Dal momento che il Blitter sa già qual è il quadrante da usare, non ha difficoltà nel tracciare la linea correttamente.

Il registro di puntamento del canale A va inizializzato con $4 * dy - 2 * dx$. Se questo valore è negativo occorre impostare il flag di segno (SIGNFLAG) in BLTCON1, altrimenti il flag va posto a zero. Il modulo di A dev'essere inizializzato a $4 * (dy - dx)$ e il modulo di B a $4 * dy$.

Il registro dati A dev'essere inizializzato a $8000. Entrambe le maschere devono avere il valore $FFFF. Il valore dello shift per il canale A deve corrispondere alla coordinata *x* del primo punto (*x1*) modulo 15.

Il registro dati B dev'essere inizializzato con la matrice richiesta per la linea ($FFFF per una linea continua). Lo shift del canale B indica il bit dal quale inizierà l'applicazione della matrice (0 rappresenta il bit meno significativo).

I registri di puntamento C e D devono puntare alla word che contiene il primo pixel della linea. Il loro modulo dev'essere uguale alla larghezza in byte del bitplane.

SRCA, SRCC e DEST in BLTCON0 devono essere impostati a 1, SRCB a 0. Dev'essere azzerato il flag OVFLAG. Se si desidera un solo pixel per ogni linea orizzontale, si deve impostare il flag ONEDOT; altrimenti il flag dev'essere azzerato.

A questo punto s'imposta la funzione logica desiderata. Il canale C rappresenta l'area originale, il canale A i bit della linea e il canale B la matrice da applicare. Perciò, per una semplice linea, la funzione più comunemente usata rimane $AB + \overline{A}C$. Per tracciare la linea in modo XOR, perché possa essere facilmente cancellata con un secondo tracciamento, si può usare la funzione $AB\overline{C} + \overline{A}C$.

L'altezza del blit equivale alla lunghezza della linea, cioè $dx + 1$. La larghezza dev'essere impostata a 2 in ogni caso. Ovviamente, il registro BLTSIZE dev'essere impostato soltanto alla fine, quando tutti gli altri registri contengono già i valori corretti.

SOMMARIO DEI REGISTRI NEL TRACCIAMENTO DI LINEE

Impostazione preliminare:

La linea va da $(x1,y1)$ a $(x2,y2)$.

```
dx = max(abs(x2 - x1), abs(y2 - y1));
dy = min(abs(x2 - x1), abs(y2 - y1));
```

Impostazione dei registri:

BLTADAT = $8000
BLTBDAT = pattern ($FFFF per una linea continua)

BLTAFWM = $FFFF
BLTALWM = $FFFF

BLTAMOD = $4 * (dy - dx)$
BLTBMOD = $4 * dy$

BLTCMOD = larghezza del bitplane in byte
BLTDMOD = larghezza del bitplane in byte

BLTAPT = $(4 * dy) - (2 * dx)$
BLTBPT = non utilizzato
BLTCPT = word contenente il primo pixel della linea
BLTDPT = word contenente il primo pixel della linea

BLTCON0, bit 15–12 = $x1$ modulo 15
BLTCON0, bit SRCA, SRCC e DEST = 1
BLTCON0, bit SRCB = 0
BLTCON0, byte LF = funzione logica richiesta

BLTCON1, bit LINEMODE = 1
BLTCON1, bit OVFLAG = 0
BLTCON1, bit 4–2 = numero dell'ottante ricavato dalla tabella
BLTCON1, bit 15-12 = bit iniziale per l'applicazione della matrice (0 = bit meno significativo)
se $((4 * dy) - (2 * dx)) < 0$:
allora BLTCON1, bit SIGNFLAG = 1
altrimenti BLTCON1, bit SIGNFLAG = 0
se la linea deve avere un solo pixel per riga:
allora BLTCON1, bit ONEDOT = 1
altrimenti BLTCON1, bit ONEDOT = 0

BLTSIZE, bit 15-6 = $dx + 1$
BLTSIZE, bit 5-0 = 2

Il registro BLTSIZE dev'essere impostato per ultimo dal momento che dà il via al blit.

Velocità del Blitter

La velocità del Blitter dipende soltanto dai canali DMA abilitati. È possibile usare un canale DMA come costante, ma, a meno che non venga abilitato non entra nel computo della velocità. Il ciclo più breve possibile del Blitter è di 4 tick, il più grande di 8. L'uso del canale A è sempre "gratuito". L'uso del canale B aggiunge due tick al totale. L'uso dei canali C o D è gratuito, ma l'uso di entrambi aggiunge altri due tick. Perciò, un ciclo di copia dati utilizzando i canali A e D impiega 4 tick, utilizzando i canali B e D 6 tick e utilizzando i canali B, C e D 8 tick. Nel tracciamento di linee, ogni pixel richiede 8 tick.

Il clock di sistema per un Amiga PAL è di 7,09 MHz (7,16 in NTSC). Il Blitter utilizza il clock di sistema. Per calcolare il tempo totale di un blit in microsecondi, escludendo dal calcolo l'impostazione dei registri e supponendo che i canali DMA siano sempre in grado di accedere alla memoria, si può usare la formula seguente:

$$t = \frac{n * H * W}{7{,}09}$$

dove t è il tempo in microsecondi, n il numero di cicli di clock (tick) per ogni ciclo del Blitter e H e W rispettivamente l'altezza e la larghezza (in word) del blit.

Per esempio, per copiare un bitplane 320 x 256 in un altro bitplane utilizzando i canali A e D (4 tick per ciclo):

$$\frac{4 * 256 * 20}{7,09} = 2889 \text{ microsecondi}$$

Le operazioni del Blitter e il DMA di sistema

Le operazioni del Blitter influiscono sulle prestazioni del resto del sistema. In questo paragrafo si esamina come il Blitter influisce sul sistema tramite la priorità del suo DMA, l'allocazione degli slot DMA, la condivisione del bus con il 68000 e l'hardware video, le operazioni del Blitter e del Copper, e i playfield di dimensioni diverse.

Il Blitter esegue tutte le sue operazioni di lettura, modifica e scrittura tramite l'uso del DMA, e condivide l'accesso alla memoria con gli altri dispositivi del sistema. Ogni dispositivo che accede alla memoria possiede un livello di priorità, che indica la sua importanza rispetto agli altri dispositivi.

La priorità più alta appartiene al DMA dei dischi, al DMA audio, al DMA video e al DMA degli sprite (in alcune circostanze il DMA video ha la precedenza su quello degli sprite). Ognuno di questi dispositivi ha un certo numero di cicli DMA assegnati a ogni linea di scansione. I cicli eventualmente non utilizzati possono essere sfruttati da altri dispositivi. La loro maggiore priorità è dovuta al fatto che cicli di DMA mancati potrebbero significare perdita di dati, rumore nell'output audio o interruzioni dell'immagine video.

Il Copper ha la priorità immediatamente successiva perché deve poter effettuare le proprie operazioni in sincronia con la scansione del video.

Le priorità più basse sono assegnate al Blitter e al 68000, in quest'ordine. Il Blitter ha una priorità più alta perché può eseguire le operazioni di copia e di tracciamento di linee molto più velocemente del 68000.

Durante ogni linea di scansione (circa 63 microsecondi), vi sono 227,5 "clock di colore", ovvero cicli di accesso alla memoria. Ogni ciclo dura approssimativamente 280 ns. Il totale di 227,5 cicli comprende sia il tempo assegnato al display sia quello non assegnato. Di questo tempo totale, 226 cicli sono disponibili per i vari dispositivi che richiedono l'accesso alla memoria.

I cicli sono distribuiti in questo modo:

4 cicli per il refresh della memoria.
3 cicli per il DMA dei dischi.
4 cicli per il DMA audio (2 byte per canale).
16 cicli per il DMA degli sprite (2 word per canale).
80 cicli per il DMA dei playfield (slot pari o dispari a seconda delle caratteristiche dello schermo).

La figura 6.9 mostra una completa linea di scansione orizzontale e l'allocazione dei cicli di clock.

Il 68000 usa soltanto i cicli pari: per metà del tempo, infatti, il microprocessore è impegnato a eseguire operazioni interne. Pertanto, assegnare al 68000 solo un ciclo ogni due non limita la sua velocità massima.

Alcune istruzioni, tuttavia, non si accordano perfettamente con questo schema, e causano la perdita di alcuni cicli. In questo caso, il 68000 deve attendere il primo ciclo disponibile per poter continuare. Ma si tratta comunque di un'eventualità poco frequente, e in realtà il 68000 funziona

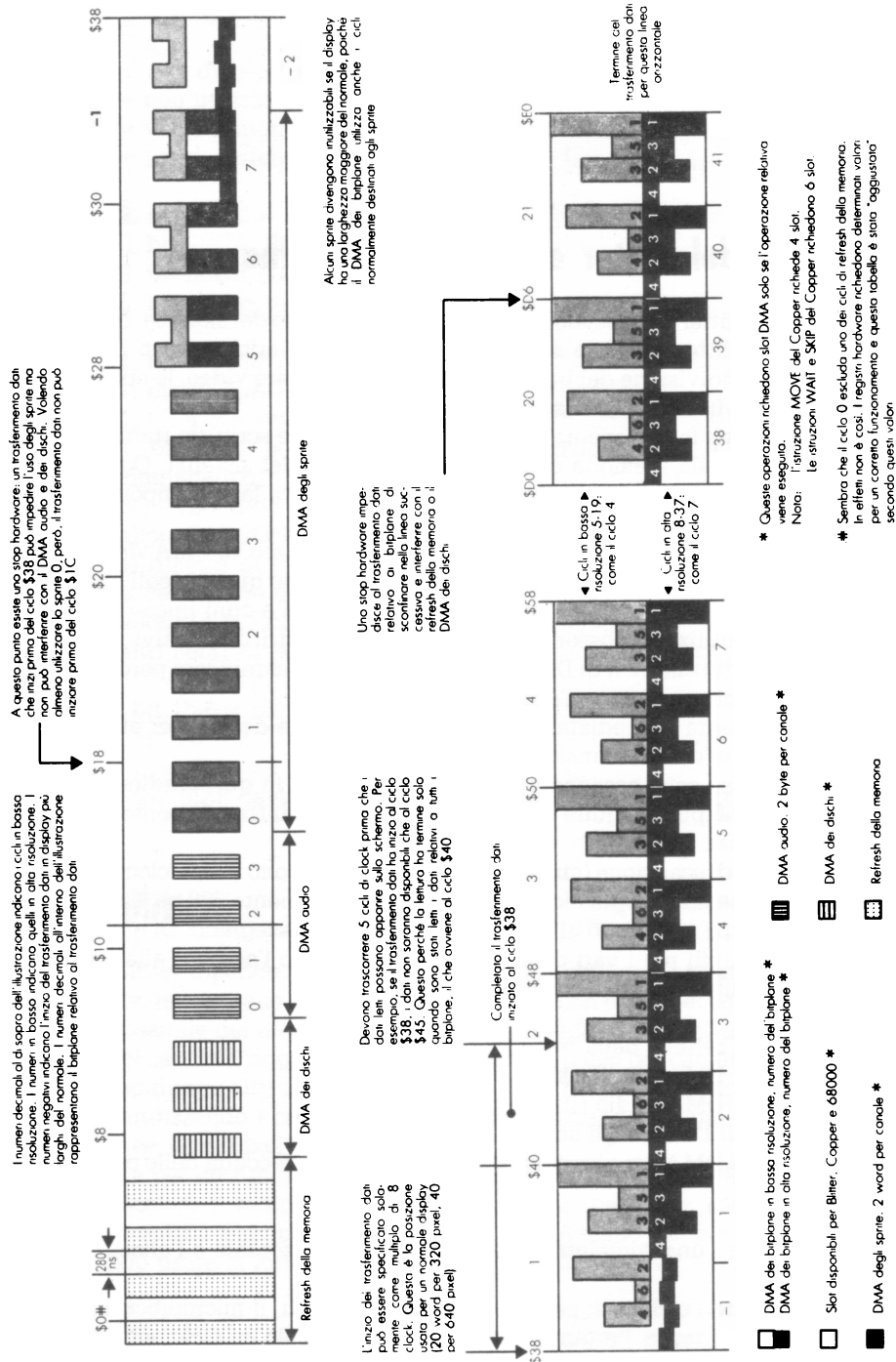



Figura 6.9: Allocazione degli slot DMA

alla massima velocità per la maggior parte del tempo.

La Figura 6.10 illustra un normale ciclo del 68000.

L'istruzione TAS del 68000 non dovrebbe essere mai utilizzata sull'Amiga. Il caratteristico ciclo indivisibile di lettura/scrittura di questa istruzione non può essere adattato all'allocazione dei cicli di DMA.

Se lo schermo video non contiene più di quattro bitplane in bassa risoluzione, al 68000 sono garantiti tutti i cicli pari di accesso alla memoria (sempre che sia in grado di utilizzarli e che sia il dispositivo con la priorità più alta). Ma se vi sono più di quattro bitplane, il DMA dei playfield inizia a sottrarre cicli al 68000.

Durante la visualizzazione di un display in bassa risoluzione formato da sei bitplane, il DMA video richiede 120 cicli per ogni linea orizzontale. Come si vede dalla Figura 6.11, ciò priva il 68000 di metà dei cicli che sarebbero disponibili in un playfield con quattro bitplane.

Specificando quattro bitplane in alta risoluzione, il DMA dei bitplane necessita di tutti i cicli disponibili per poter trasferire le 160 word necessarie. In una simile condizione, il 68000 (e anche il Blitter e il Copper) possono accedere alla memoria soltanto durante gli intervalli di blanking, verticale e orizzontale.

Ogni linea di un normale playfield contiene 320 o 640 pixel (a seconda della risoluzione) che corrispondono a un trasferimento di 20 o 40 word per riga. Volendo muovere orizzontalmente un playfield occorre una word in più.

Le dimensioni della finestra video sono modificabili, e il DMA dei bitplane ha la precedenza su quello degli sprite. Come si vede nella Figura 6.9, display di dimensioni maggiori del consueto possono impedire la visualizzazione degli sprite aventi numero più alto, specialmente nel caso di scroll del video.

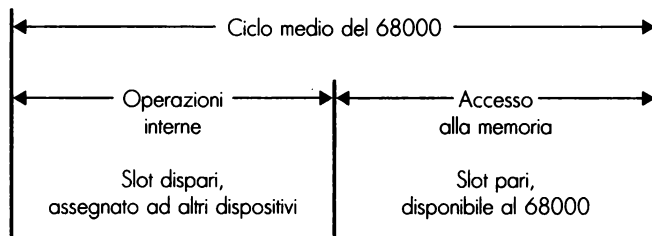



Figura 6.10: Ciclo normale del 68000

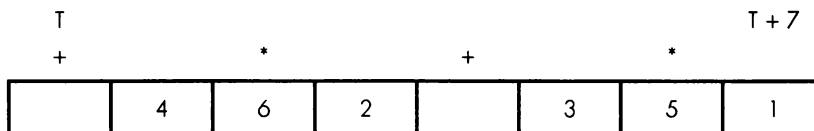



Figura 6.11: Slot usati da un display a sei bitplane

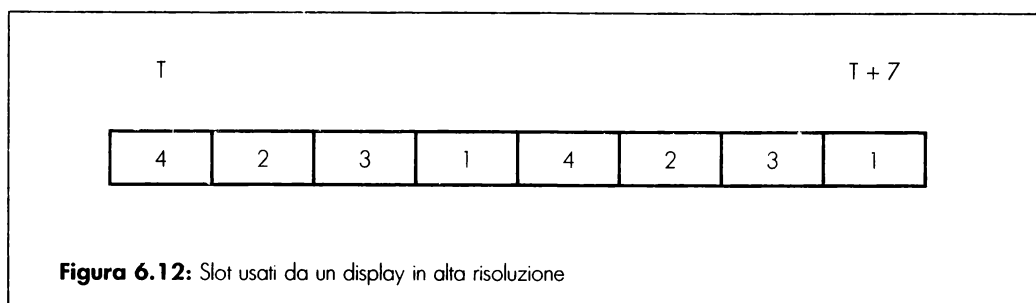



Figura 6.12: Slot usati da un display in alta risoluzione

Come abbiamo già detto, il Blitter in genere ha una priorità maggiore del 68000 per l'uso dei cicli di DMA. Vi sono tuttavia alcuni casi in cui il Blitter e il 68000 possono condividere cicli di memoria. Se ne ha l'opportunità, il Blitter utilizza ogni ciclo disponibile, ed è per questo che il video, l'audio e i dischi (che non devono essere bloccati) hanno la precedenza sul Blitter. A seconda dell'impostazione del bit DMAF_BLITHOG nel registro DMACON, il 68000 può avere accesso al bus o essere completamente bloccato.

Se DMAF_BLITHOG è a 1, il Blitter utilizza ogni ciclo disponibile del bus. Quindi, in teoria, tutti i cicli.

Se invece DMAF_BLITHOG è a 0, quando tre richieste consecutive di accesso alla memoria del 68000 non vengono soddisfatte, il Blitter viene forzato a cedere il bus per un ciclo.

Diagramma a blocchi del Blitter

- La Figura 6.13 mostra i blocchi logici in cui viene divisa una singola operazione su 16 bit del Blitter.

- Nell'angolo superiore sinistro si vede come vengono applicate le maschere alla sorgente A. Nel caso di un blit di una sola word di larghezza, vengono applicate entrambe le maschere.

- I due shifter mostrano come i 16 bit di dati vengano estratti da una specifica posizione all'interno del registro a 32 bit, in base ai valori indicati in BLTCON0 e BLTCON1.

- Il generatore di minterm mostra come i bit di selezione dei minterm permettano o inibiscano l'uso di uno specifico minterm.

- Si vede inoltre in che modo l'operazione di riempimento di aree operi con i dati generati dalla combinazione dei minterm. L'operazione può essere quindi associata a qualunque funzione logica.

- Nella parte inferiore dell'immagine si vede come si può impedire che la destinazione venga scritta (tramite un bit di controllo del Blitter).

- Non è invece presente in questo diagramma la logica per la rilevazione dell'azzeramento, che controlla ogni bit generato per la destinazione. Se viene generato anche un solo bit a 1, il bit di rilevazione dello zero viene impostato a FALSE (0).

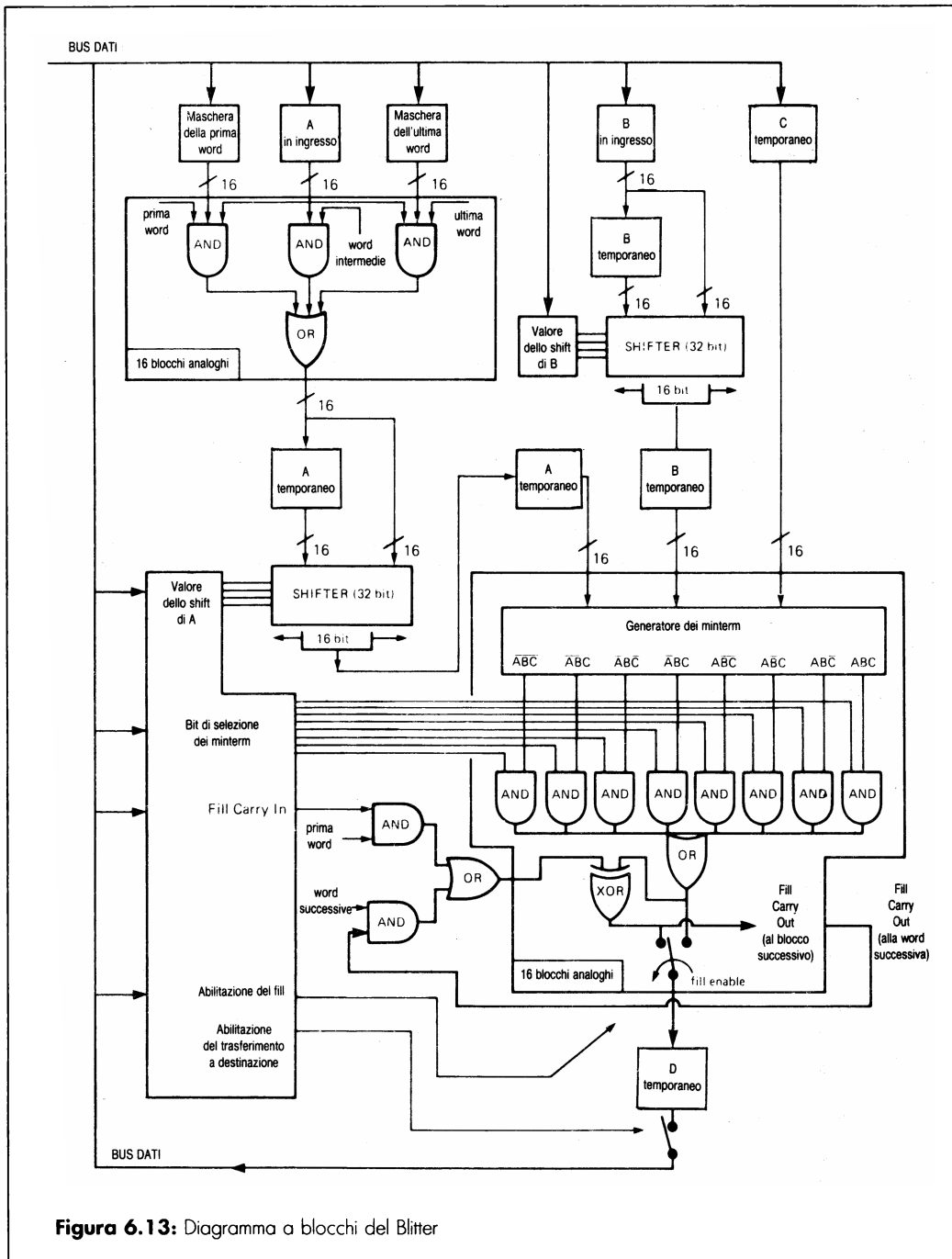



Figura 6.13: Diagramma a blocchi del Blitter

Ciò che si deve ricordare

Ecco una lista di alcuni punti chiave che conviene avere ben presenti durante la programmazione del Blitter.

- Impostare sempre BLTSIZE per ultimo: la scrittura di questo registro dà il via al blit.
- Puntatori e moduli sono espressi in byte, la larghezza in word e l'altezza in pixel. Il bit meno significativo dei puntatori e dei moduli viene ignorato.
- L'ordine delle operazioni è: mascheratura, scorrimento, combinazione logica delle sorgenti, riempimento di aree e impostazione del flag di zero.
- Nel modo ascendente, il Blitter incrementa i puntatori, somma i moduli ed esegue scorrimenti verso destra.
- Nel modo discendente, il Blitter decrementa i puntatori, sottrae i moduli ed esegue scorrimenti verso sinistra.
- Il riempimento di aree funziona correttamente soltanto nel modo discendente.
- Controllare BLTDONE prima di riutilizzare i registri del Blitter o il risultato di un blit.
- Per i dati, lo shift avviene subito dopo il caricamento.

ESEMPIO: ClearMem

```
;
;Esempio: azzeramento di un'area di memoria
;
        include "exec/types.i"
        include "hardware/custom.i"
        include "hardware/dmabits.i"
        include "hardware/blit.i"
        include "hardware/hw_examples.i"

        xref _custom
;
;Attende il termine del blit precedente.
;
waitblit:
        btst.b  #DMAB_BLTDONE-8,DMACONR(a1)
waitblit2:
        btst.b  #DMAB_BLTDONE-8,DMACONR(a1)
        bne.s   waitblit2
        rts
;
;Questa routine sfrutta un effetto collaterale del Blitter:
;quando un blit e' completato, i puntatori del Blitter puntano
```

```
;alla word successiva al blit.
;
;Quando questa routine restituisce il controllo, l'ultimo blit
;potrebbe non essere ancora terminato. Si deve attendere tramite
;waitblit che i dati siano veramente azzerati.
;
;a0 = puntatore alla prima word da azzerare
;d0 = numero di byte da azzerare (pari)
;
      xdef    _clearmem
clearmem:
      lea     _custom,a1                  ;Puntatore ai chip custom
      bsr     waitblit                    ;Lascia finire il blit precedente
      move.l  a0,BLTDPT(a1)               ;Destinazione
      clr.w   BLTDMOD(a1)                 ;Modulo D = 0
      asr.l   #1,d0                       ;Trasforma d0 nel numero di word
      clr.w   BLTCON1(a1)                 ;Nessun modo speciale
      move.w  #DEST,BLTCON0(1)            ;Abilita solo la destinazione
;
;Prima i blit piu' piccoli.
;
      moveq   #$3F,d1                     ;Maschera per le ultime 64 word
      and.w   d0,d1
      beq.s   seguito                     ;Passa al blocco principale
      sub.l   d1,d0                       ;Altrimenti sottrae il resto
      or.l    #$40,d1                     ;Altezza = 1, larghezza = n
      move.w  d1,BLTSIZE(a1)              ;Da' inizio al blit
;
;Ed ecco il resto, in blocchi di 128K
;
seguito:
      move.w  #$FFC0,d1                   ;Esamina i bit piu' significativi
      and.w   d0,d1                       ;Estrae 10 bit
      beq.s   seguito2                    ;Altro da fare?
      sub.l   d1,d0                       ;Sottrae le word corrispondenti
      bsr     waitblit                    ;Lascia finire il blit precedente
      move.w  d0,BLTSIZE(a1)              ;un altro blit
seguito2:
      swap    d0                          ;C'e' dell'altro?
      beq.s   finito                      ;No
      clr.w   d1                          ;Un blit di 1024 x 64 word (128K)
continua:
      bsr     waitblit                    ;Finisce il blit precedente
      move.w  d1,BLTSIZE(a1)              ;Ancora un blit
      subq.w  #1,d0                       ;Ancora?
      bne.s   continua                    ;Ripete il loop
finito:
      rts

      end                                 ;Finito, blit ancora in corso.
```

ESEMPIO: SimpleLine

```
;
;In questo esempio si sfrutta il Blitter per tracciare una linea
;La linea e' continua e viene tracciata con una semplice funzione
;di or logico.
;
;Input: d0=x1 d1=y1 d2=x2 d3=y2 d4=larghezza a0=aptr
        include "exec/types.i"
        include "hardware/custom.i"
        include "hardware/dmabits.i"
        include "hardware/blit.i"
        include "hardware/hw_examples.i"

        xref    _custom

        xdef    _simpleline

simpleline:
        lea     _custom,a1                  ;base dei chip custom
        sub.w   d0,d2                       ;dx
        bmi.s   xneg                        ;se dx<0 ottante = 3, 4, 5 o 6
        sub.w   d1,d3                       ;dy      ottante = 1, 2, 7 o 8
        bmi.s   yneg                        ;se dy<0 ottante = 7 o 8
        cmp.w   d3,d2                       ;dy>dx?  ottante = 1 o 2
        bmi.s   ygtx                        ;se y>x  ottante = 2
        moveq   #OCTANT1+LINEMODE,d5        ;altrimenti ottante = 1
        bra.s   linea
ygtx:
        exg     d2,d3                       ;dx dev'essere maggiore di dy
        moveq   #OCTANT2+LINEMODE,d5        ;ottante = 2
        bra.s   linea
yneg:
        neg.w   d3                          ;abs(dy)
        cmp.w   d3,d2                       ;dy>dx?  ottante = 7 o 8
        bmi.s   ynygtx                      ;se y>x  ottante = 7
        moveq   #OCTANT8+LINEMODE,d5        ;altrimenti ottante = 8
        bra.s   linea
ynygtx:
        exg     d2,d3                       ;dx dev'essere maggiore di dy
        moveq   #OCTANT7+LINEMODE,d5        ;ottante = 7
        bra.s   linea
xneg:
        neg.w   d2                          ;abs(dx) ottante = 3, 4, 5 o 6
        sub.w   d1,d3                       ;dy
        bmi.s   xyneg                       ;se dy<0 ottante = 5 o 6
        cmp.w   d3,d2                       ;dy>dx?  ottante = 3 o 4
        bmi.s   xnygtx                      ;se y>x  ottante = 3
        moveq   #OCTANT4+LINEMODE,d5        ;altrimenti ottante = 4
        bra.s   linea
```

```
xnygtx:
      exg     d2,d3                         ;dx dev'essere maggiore di dy
      moveq   #OCTANT3+LINEMODE,d5          ;ottante = 3
      bra.s   linea
xyneg:
      neg.w   d3                            ;abs(dy) ottante = 5 o 6
      cmp.w   d3,d2                         ;dy>dx?
      bmi.s   xynygtx                       ;se si'   ottante = 6
      moveq   #OCTANT5+LINEMODE,d5          ;altrimenti ottante = 5
      bra.s   linea
xynygtx:
      exg     d2,d3                         ;dx dev'essere maggiore di dy
      moveq   #OCTANT6+LINEMODE,d5          ;ottante = 6
linea:
      mulu.w  d4,d1                         ;y1 * larghezza
      ror.l   #4,d0                         ;d0 >> 4 (4 lsb in d0h)
      add.w   d0,d0                         ;d0 * 2
      adda.w  d0,a0                         ;ptr += (x1 >> 3)
      adda.l  d1,a0                         ;ptr += y1 * larghezza
      swap    d0                            ;4 lsb di x1 in d0l
      or.w    #$BFA,d0                      ;or con USEA, USEC e USED, LF = A + C
      lsl.w   #2,d3                         ;dx = 4 * dx
      add.w   d2,d2                         ;dy = 2 * dy
      move.w  d2,d1                         ;preparazione di BLTSIZE
      lsl.w   #5,d1                         ;5 posizioni a sinistra
      add.w   #$42,d1                       ;larghezza += 2, altezza += 1
      btst    #DMAB_BLTDONE-8,DMACONR(a1)   ;controllo di sicurezza
waitblit:
      btst    #DMAB_BLTDONE-8,DMACONR(a1)   ;attende il Blitter
      bne.s   waitblit
      move.w  d3,BLTBMOD(a1)                ;modulo B = 4 * dy
      sub.w   d2,d3
      ext.l   d3
      move.l  d3,BLTAPT(a1)                 ;puntatore A = 4 * dy - 2 * dx
      bpl.s   traccialinea
      or.w    #SIGNFLAG,d5                  ;bit di segno in BLTCON1
traccialinea:
      move.w  d0,BLTCON0(a1)
      move.w  d5,BLTCON1(a1)
      move.w  d4,BLTCMOD(a1)                ;modulo C = larghezza del bitplane
      move.w  d4,BLTDMOD(a1)                ;modulo D = larghezza del bitplane
      sub.w   d2,d3
      move.w  d3,BLTAMOD(a1)                ;modulo A = 4 * dy - 4 * dx
      move.w  #$8000,BLTADAT(a1)
      moveq   #-1,d5                        ;d5 = $FFFFFFFF
      move.l  d5,BLTAFWM(a1)                ;entrambe le maschere in un colpo solo
      move.l  a0,BLTCPT(a1)                 ;word contenente il primo pixel
      move.l  a0,BLTDPT(a1)
      move.w  d1,BLTSIZE(a1)                ;inizio del blit
      rts                                   ;ritorno, blit ancora in corso
      end
```

ESEMPIO: RotateBits

```
;
;Questo esempio prende una singola riga di un bitplane e la
;"ruota" all'interno di un array di word. Per ottenere questo
;effetto, viene usato il modo per il tracciamento di linee
;insieme a una particolare matrice.
;
;Input: d0 contiene il numero di word di ogni riga. d1 contiene
;il numero del bit da impostare (0...15). a0 contiene un
;puntatore ai dati del bitplane e a1 un puntatore all'array
;da riempire; l'array dev'essere lungo almeno d0 * 16 word.
;
      include "exec/types.i"
      include "hardware/custom.i"
      include "hardware/dmabits.i"
      include "hardware/blit.i"
      include "hardware/hw_examples.i"

      xref    _custom

      xdef    _rotatebits

rotatebits:
      lea     _custom,a2
      tst     d0                        ;Se non vi sono word ritorna subito
      beq.s   fatto
      lea     DMACONR(a2),a3
      moveq   #DMAB_BLTDONE-8,d2
      btst    d2,(a3)                   ;Controlla se il Blitter ha finito
wait1:
      btst    d2,(a3)                   ;Controlla di nuovo
      bne.s   wait1
      moveq   #-30,d3                   ;linea: aptr = 4Y - 2X, Y = 0, X = 15
      move.l  d3,BLTAPT(a2)
      move.w  #-60,BLTAMOD(a2)          ;modulo A = 4Y - 4X
      clr.w   BLTBMOD(a2)               ;modulo B = 4Y
      move.w  #2,BLTCMOD(a2)            ;modulo C = larghezza (2)
      move.w  #2,BLTDMOD(a2)            ;idem
      ror.w   $4,d1                     ;numero del bit
      and.w   #$F000,d1                 ;maschera il resto
      or.w    #$BCA,d1                  ;USEA, USEC, USED, LF=AB+
AC
      move.w  d1,BLTCON0(a2)
      move.w  #$F049,BLTCON1(a2)        ;BSHIFT = 15, SGN, LINE
      move.w  #$8000,BLTADAT(a2)        ;dati per una linea
      move.w  #$FFFF,BLTAFWM(a2)
      move.w  #$FFFF,BLTALWM(a2)
      move.l  a1,BLTCPT(a2)             ;destinazione
      move.l  a1,BLTDPT(a2)
      lea     BLTBDAT(a2),a4
```

```
        lea     BLTSIZE(a2),a5
        move.w  #$402,d1                    ;larghezza = 2, altezza = 16
        move.w  (a0)+,d3                    ;preleva la word successiva
        bra.s   loop                        ;entra nel loop
ripeti:
        move.w  (a0)+,d3                    ;un'altra word
        btst    d2,(a3)                     ;blit terminato?
wait2:
        btst    d2,(a3)                     ;blit terminato?
        bne.s   wait2
loop:
        move.w  d3,(a4)                     ;utilizza la word come pattern
        move.w  d1,(a5)                     ;inizia il blit
        subq.w  #1,d0                       ;ultima word?
        bne.s   ripeti
fatto:
        rts
        end
```

7 HARDWARE DI CONTROLLO

Introduzione

Questo capitolo descrive l'hardware di controllo del sistema Amiga, che comprende i seguenti elementi:

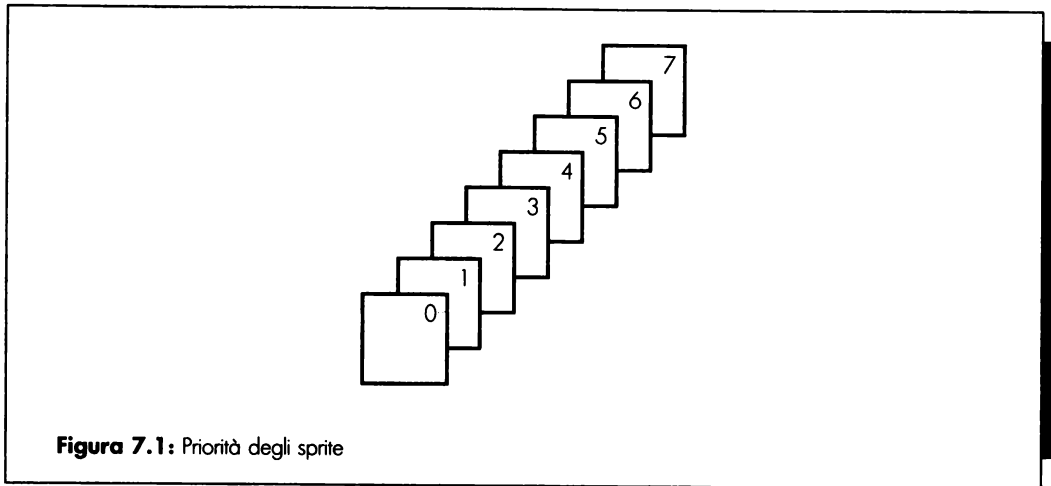
- Priorità relative dei playfield e degli sprite.
- Rilevazione delle collisioni fra oggetti video.
- Controllo del DMA.
- Controllo degli interrupt.
- Procedure di accensione e di reset.

Priorità video

È possibile controllare la priorità di vari oggetti sul video al fine di dare l'illusione della tridimensionalità. Il paragrafo seguente tratta la priorità relativa di playfield e sprite.

PRIORITÀ DEGLI SPRITE

Non è possibile modificare la priorità relativa degli sprite. La priorità più alta corrisponde sempre allo sprite avente numero inferiore, come mostra la Figura 7 1. Ogni quadrato rappresenta lo sprite identificato dal numero al suo interno.

Figura 7.1: Priorità degli sprite

COME SONO RAGGRUPPATI GLI SPRITE

Per quanto riguarda la priorità rispetto a playfield e collisioni, gli sprite vengono raggruppati in quattro coppie:

Sprite 0 e 1
Sprite 2 e 3
Sprite 4 e 5
Sprite 6 e 7

IL SIGNIFICATO DELLA PRIORITÀ VIDEO

Per capire il concetto di priorità video, immaginiamo che quattro dita di una mano rappresentino le quattro coppie di sprite, e due dita dell'altra mano rappresentino i playfield 1 e 2. Così come è impossibile scambiare fra loro le dita di una mano, è impossibile anche modificare le priorità relative degli sprite. Tuttavia, le due dita dell'altra mano possono essere inserite in diverse posizioni tra le dita della prima, e così possono comportarsi i playfield rispetto agli sprite. La situazione è illustrata nella Figura 7.2.

Sono quindi cinque le posizioni possibili per ognuno dei due playfield. Per esempio, è possibile collocare il playfield 1 al di sopra degli sprite 0 e 1 **(0)**, tra gli sprite 0 e 1 e gli sprite 2 e 3 **(1)**, tra gli sprite 2 e 3 e gli sprite 4 e 5 **(2)**, tra gli sprite 4 e 5 e gli sprite 6 e 7 **(3)**, o sotto gli sprite 6 e 7 **(4)**. Le stesse possibilità esistono per il playfield 2.

I numeri tra parentesi indicano il valore da utilizzare per selezionare la priorità dei playfield (si veda il paragrafo seguente).

È inoltre possibile controllare la priorità dei playfield 1 e 2 l'uno rispetto all'altro. Ciò permette ulteriori scelte nell'impostazione delle priorità.

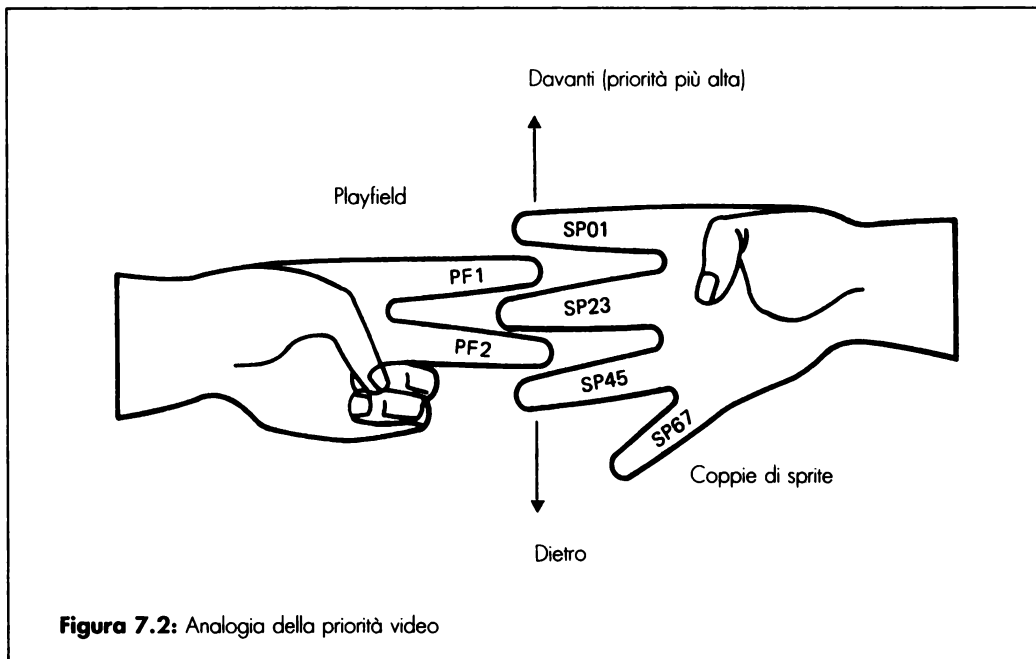



Figura 7.2: Analogia della priorità video

IMPOSTAZIONE DEL REGISTRO DI CONTROLLO DELLA PRIORITÀ

Questo registro permette di definire le priorità relative dei vari oggetti video. In genere il playfield 1 appare davanti al playfield 2. Il bit PF2PRI inverte questa situazione, dando la preminenza al playfield 2. Il significato dei bit del registro BPLCON2 è rappresentato nella tavola seguente.

Tavola 7.1: Significato dei bit di BPLCON2

Bit	Nome	Funzione
15-7		Non utilizzati (azzerare)
6	PF2PRI	Priorità al playfield 2
5-3	PF2P2-PF2P0	Posizione del playfield 2 rispetto agli sprite
2-0	PF1P2-PF1P0	Posizione del playfield 1 rispetto agli sprite

Il valore binario assegnato ai bit PF1P2-PF1P0 determina la posizione del playfield 1 rispetto agli sprite, come si vede nella Tavola 7.2.

I bit PF2P2-PF2P0 (i bit 5-3) sono quelli che vengono utilizzati per un playfield normale (cioè non in modo dual-playfield).

Tavola 7.2: Priorità dei playfield in relazione ai bit PF1P2-PF1P0

Valore	**Posizione (in ordine d'importanza decrescente)**				
000	PF1	SP01	SP23	SP45	SP67
001	SP01	PF1	SP23	SP45	SP67
010	SP01	SP23	PF1	SP45	SP67
011	SP01	SP23	SP45	PF1	SP67
100	SP01	SP23	SP45	SP67	PF1

In questa tavola, PF1 sta a indicare il playfield 1, SP01 il gruppo formato dagli sprite 0 e 1, SP23 quello formato dagli sprite 2 e 3, SP45 quello formato dagli sprite 4 e 5 e SP67 quello formato dagli sprite 6 e 7.

Parallelamente, i bit PF2P2-PF2P0 consentono di determinare la posizione del playfield 2 rispetto agli sprite. Tuttavia, è il bit PF2PRI a determinare quale dei due playfield apparirà di fronte all'altro. Ecco un possibile esempio d'impostazione dei bit di BPLCON2 che dà origine a una situazione un po' insolita:

BIT	15-7	PF2PRI	PF2P2-0	PF1P2-0
Valore	0	1	010	000

Il risultato è seguente ordine di priorità tra sprite e playfield:

PF1	SP01	SP23	PF2	SP45	SP67

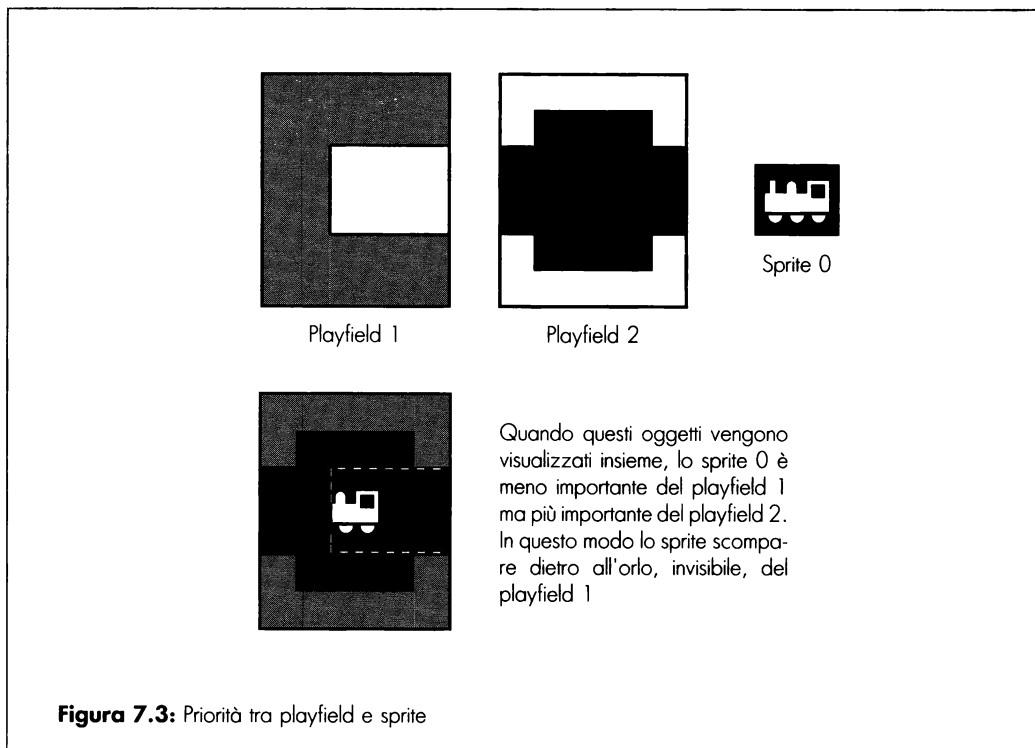
In altre parole, il playfield 1 appare davanti a tutti gli sprite, mentre gli sprite 0-3 sono davanti al playfield 2. Tuttavia, il playfield 2 appare davanti al playfield 1 nelle zone dove i due playfield si sovrappongono (purché il playfield 2 non sia bloccato dagli sprite 0-3).

La figura seguente mostra una possibile conseguenza di questa impostazione. Lo sprite 0 può muoversi al di sopra del playfield 2, ma quando attraversa il playfield 1 viene coperto. Il risultato è un insolito effetto video che fa scomparire lo sprite quando oltrepassa un invisibile limite sullo schermo.

Rilevazione delle collisioni

Si può utilizzare l'hardware per rilevare collisioni tra due gruppi di sprite, fra un gruppo di sprite e un playfield, fra due playfield o fra una qualunque combinazione di queste entità.

Il primo tipo di collisione è in genere il più utilizzato nei videogame (per esempio per sapere se un missile ha colpito un bersaglio). Il secondo tipo è invece utilizzabile per mantenere un oggetto in movimento all'interno di determinati limiti sullo schermo. Il terzo tipo di collisione permette di definire parti dei playfield come singoli oggetti, mossi, per esempio, tramite il Blitter (questa tecnica prende il nome di "animazione di playfield"). Se uno dei playfield viene usato per definire lo sfondo dell'area di gioco e l'altro per definire oggetti in movimento (in aggiunta agli sprite), è quindi possibile rilevare le collisioni di questi oggetti con gli sprite o con il primo playfield.

Figura 7.3: Priorità tra playfield e sprite

COME SI DETERMINANO LE COLLISIONI

L'output video è formato dai dati provenienti dai bitplane e dagli sprite, combinati in un unico flusso di dati. Il colore dei pixel dello schermo è dato dall'oggetto a priorità più alta che si trova in quel punto. Viene rilevata una collisione quando due o più oggetti occupano lo stesso pixel, e viene quindi impostato un particolare bit nel registro di collisione.

COME S'INTERPRETANO I DATI RELATIVI ALLE COLLISIONI

Il registro contenente i dati relativi alle collisioni, CLXDAT, è a sola lettura, e il suo contenuto viene automaticamente azzerato dopo ogni lettura. Il significato dei suoi bit viene riassunto nella tavola seguente.

Tavola 7.3: Bit di CLXDAT

Bit	Collisione registrata
15	Non utilizzato
14	Sprite 4 (o 5) con lo sprite 6 (o 7)
13	Sprite 2 (o 3) con lo sprite 6 (o 7)
12	Sprite 2 (o 3) con lo sprite 4 (o 5)
11	Sprite 0 (o 1) con lo sprite 6 (o 7)

10	Sprite 0 (o 1) con lo sprite 4 (o 5)
9	Sprite 0 (o 1) con lo sprite 2 (o 3)
8	Bitplane pari con lo sprite 6 (o 7)
7	Bitplane pari con lo sprite 4 (o 5)
6	Bitplane pari con lo sprite 2 (o 3)
5	Bitplane pari con lo sprite 0 (o 1)
4	Bitplane dispari con lo sprite 6 (o 7)
3	Bitplane dispari con lo sprite 4 (o 5)
2	Bitplane dispari con lo sprite 2 (o 3)
1	Bitplane dispari con lo sprite 0 (o 1)
0	Bitplane pari con bitplane dispari

I numeri tra parentesi nella tavola si riferiscono a collisioni di cui è possibile abilitare o meno la rilevazione. Il registro di controllo, di cui parleremo in seguito, permette infatti di ignorare (eventualmente) le collisioni con gli sprite dispari.
Si noti che la rilevazione delle collisioni non cambia utilizzando il modo dual-playfield, poiché dipende soltanto dallo stato dei bit dei bitplane pari o dispari. Il registro di controllo delle collisioni consente di specificare l'uso esatto dei bitplane in questa rilevazione.

COME SI CONTROLLA LA RILEVAZIONE DELLE COLLISIONI

I bit del registro CLXCON specificano alcune caratteristiche della rilevazione di collisioni. Il loro significato è riportato nella tavola che segue.

Tavola 7.4: Bit di CLXCON

Bit	Nome	Funzione
15	ENSP7	Abilita lo sprite 7 (OR con lo sprite 6)
14	ENSP5	Abilita lo sprite 5 (OR con lo sprite 4)
13	ENSP3	Abilita lo sprite 3 (OR con lo sprite 2)
12	ENSP1	Abilita lo sprite 1 (OR con lo sprite 0)
11	ENBP6	Abilita il bitplane 6
10	ENBP5	Abilita il bitplane 5
9	ENBP4	Abilita il bitplane 4
8	ENBP3	Abilita il bitplane 3
7	ENBP2	Abilita il bitplane 2
6	ENBP1	Abilita il bitplane 1
5	MVBP6	Valore di confronto per la collisione con il bitplane 6
4	MVBP5	Valore di confronto per la collisione con il bitplane 5
3	MVBP4	Valore di confronto per la collisione con il bitplane 4
2	MVBP3	Valore di confronto per la collisione con il bitplane 3
1	MVBP2	Valore di confronto per la collisione con il bitplane 2
0	MVBP1	Valore di confronto per la collisione con il bitplane 1

I bit 15-12 consentono di stabilire se le collisioni con una coppia di sprite devono riguardare anche lo sprite dispari. Gli sprite pari, invece, vengono sempre considerati. I bit 11-6 consentono di includere o escludere specifici bitplane dalla rilevazione. I bit 5-0 consentono di definire la polarità (la condizione di vero/falso) che devono avere i bit perché sia registrata una collisione.

Per esempio, potrebbe essere necessario rilevare una collisione solo quando l'oggetto si sovrappone a "qualcosa di verde" o "qualcosa di blu". Questa caratteristica, insieme ai bit di abilitazione dei bitplane, consente di specificare quali bit, e con quale polarità devono dare origine a una collisione.

Questo registro è a sola lettura. Se vengono esclusi (disabilitati) tutti i bitplane, ci si trova in una situazione di collisione perenne.

Rilevazione della posizione del pennello elettronico

A volte si devono sincronizzare le operazioni del 68000 con la posizione del pennello elettronico che crea l'immagine video. In alcuni casi potrebbe essere necessario, per esempio, modificare una certa area dello schermo *dopo* che il pennello elettronico ha generato sul monitor la relativa immagine.

A questo scopo è disponibile un registro che contiene il valore corrispondente alla posizione raggiunta dal pennello elettronico.

Si noti che il Copper compie alcune operazioni automatiche sui registri in base a questa posizione. Si consulti il Capitolo 2 per ulteriori informazioni.

Nell'uso di una penna ottica, questo stesso indirizzo viene usato per ricavare la posizione della penna. Si veda anche il Capitolo 8.

USO DEL CONTATORE DELLA POSIZIONE DEL PENNELLO ELETTRONICO

Vi sono quattro indirizzi che corrispondono ai registri di posizione del pennello elettronico. Il loro uso è descritto nella Tavola 7.5.

Tavola 7.5: Contenuto dei registri di controllo della posizione del pennello elettronico

VPOSR	*a sola lettura*	Bit più significativo della posizione verticale (V8) e identificatore del tipo di quadro video.
	Bit 15	LOF (Bit di quadro lungo). Usato per inizializzare display in modo interlace.
	Bit 14-1	Non utilizzati.
	Bit 0	Bit più significativo della posizione verticale (V8). Permette alla posizione di assumere valori maggiori di 255 sugli Amiga di tipo PAL (massimo = 313).
VHPOSR	*a sola lettura*	Posizione orizzontale e verticale del pennello elettronico (o della penna ottica).
	Bit 15-8	Bit V7-V0 della posizione verticale.
	Bit 7-0	Bit H8-H1 della posizione orizzontale. La risoluzione equivale a 1/160 della larghezza dello schermo.

VPOSW	*a sola scrittura*	Bit come in VPOSR.
VHPOSW	*a sola scrittura*	Bit come in VHPOSR.

Come di consueto, le coppie VPOSR, VHPOSR e VPOSW, VHPOSW possono essere lette e scritte come long word, tramite l'indirizzo VPOSR o VPOSW.

Interrupt

L'Amiga prevede l'utilizzazione dell'intera gamma di interrupt disponibili con il 68000. I vari tipi di interrupt generati dall'hardware sono controllati dai chip custom che li traducono in sei dei sette livelli di interrupt del 68000.

INTERRUPT NON MASCHERABILE

L'interrupt di livello 7 corrisponde all'interrupt non mascherabile, e l'attuale generazione di Amiga non è in grado di generarlo. Le linee di interrupt del 68000, da IPL2 a IPL0, sono disponibili sul connettore di espansione e possono essere utilizzate per generare interrupt di livello 7 a scopo di debug.

INTERRUPT MASCHERABILI

Vengono generati all'interno dell'Amiga gli interrupt di livello 1-6. I registri di controllo contenuti nei chip custom consentono di mascherare alcune delle loro sorgenti per impedire la generazione di interrupt di determinati livelli.

INTERFACCIA CON IL SISTEMA DEGLI INTERRUPT

Il sistema è stato progettato per poter gestire tutti i livelli di interrupt hardware convertiti nei livelli da 1 a 6. Una serie separata di linee di input, denominate INT2* e INT6* (* indica un segnale attivo basso), sono state portate al connettore di espansione per l'uso da parte di hardware esterno. Sono conosciute come le linee di interrupt esterno, rispettivamente di basso livello (2) e di alto livello (6).
Si raccomanda di utilizzare queste linee per poter approfittare degli handler di sistema, e di non generare gli interrupt direttamente tramite le linee IPL2-IPL0.

REGISTRI DI CONTROLLO DEGLI INTERRUPT

Per gli interrupt vi sono due registri di controllo: un registro di abilitazione (maschera), e un registro di richiesta (registro di stato). Ognuno di essi possiede un indirizzo per la lettura e uno per la scrittura.

I nomi degli indirizzi sono:

INTENA
Maschera di abilitazione degli interrupt - *a sola scrittura*.

INTENAR
Maschera di abilitazione degli interrupt - *a sola lettura*.

INTREQ
Richiesta di interrupt (stato) - *a sola scrittura*. Usato dal 68000 per forzare un determinato interrupt (interrupt software). Utilizzato anche per azzerare la richiesta quando l'interrupt è stato completato.

INTREQR
Richiesta di interrupt (stato) - *a sola lettura*. Contiene i bit che definiscono quali dispositivi stanno richiedendo un interrupt.

La posizione dei bit nel registro di richiesta corrisponde esattamente alla posizione nel registro di abilitazione. La sola differenza fra i registri a sola scrittura e quelli a sola lettura è che in questi ultimi il bit 15 non ha significato.

IMPOSTAZIONE E AZZERAMENTO DEI BIT

Nei paragrafi che seguono sono riportati i significati dei bit dei registri di controllo.

Impostazione e azzeramento

I registri riguardanti gli interrupt, così come quelli riguardanti il DMA, utilizzano un sistema speciale per l'impostazione e l'azzeramemto dei singoli bit: utilizzano il bit 15 dello stesso registro, detto bit SET/CLR.

Volendo impostare un bit (portarlo a 1), occorre collocare un 1 nella posizione desiderata *e un 1 nella posizione 15*.

Volendo azzerare un bit (portarlo a 0), occorre porre un 1 nella posizione desiderata *e uno 0 nella posizione 15*.

Le posizioni 14-0 selezionano i bit da modificare. Ogni bit da modificare dev'essere impostato a 1. Come i bit selezionati vengano modificati dipende dal bit 15: verranno posti a 1 o a 0 a seconda del suo stato. I bit a 0 nelle posizioni 14-0 non vengono mai modificati da un'operazione di scrittura. Volendo portare alcuni bit a 1 e altri a 0 sarà quindi necessario ricorrere a due operazioni distinte: una di impostazione e una di azzeramento.

Bit master di abilitazione degli interrupt

Il bit 14 dei registri di interrupt è il bit "master" di abilitazione degli interrupt. Quando questo bit è a zero, disabilita *tutti* gli altri interrupt. Può essere necessario ricorrere a questo bit per disabilitare temporaneamente gli interrupt in sezioni particolarmente delicate di un programma.

Questo bit serve soltanto per lo scopo illustrato: di per sé, non dà origine a nessuna richiesta di interrupt.

Interrupt esterni

I bit 13 e 3 dei registri di interrupt sono riservati agli interrupt esterni. Il bit 13, EXTER, va a 1 quando la linea INT6* diventa uno 0 logico. Questo bit genera un interrupt di livello 6. Il bit 3, PORTS, diventa un 1 quando la linea INT2* diventa uno 0 logico. Questo bit genera un interrupt di livello 2.

Interrupt di vertical blanking

Il bit 5, VERTB, genera un interrupt alla linea 0 di ogni quadro video, ovvero all'inizio dell'intervallo di vertical blanking. Il sistema deve in genere eseguire molti compiti in questo intervallo. Per esempio aggiornare i vari registri di puntamento e le liste di istruzioni del Copper.

La durata minima dell'intervallo di vertical blanking è di 25 linee (termina cioè alla linea 25) in un sistema PAL, e di 20 linee in un sistema NTSC. Questo intervallo minimo si può prolungare agendo sul valore presente nel registro DIWSTRT, che controlla l'inizio della finestra video. Si consulti il Capitolo 3 per approfondire l'uso di DIWSTRT.

Se è necessario disporre di un tempo ancora maggiore durante l'intervallo di vertical blanking, si può usare il Copper per generare un interrupt di livello 3, facendo in modo che questo interrupt abbia luogo subito dopo l'ultima linea dello schermo (definita tramite il registro DIWSTOP).

Interrupt del Copper

Il bit 4, COPER, viene usato dal Copper per generare un interrupt di livello 3. Il Copper, in realtà, può modificare *qualsiasi* bit di questi registri. Tuttavia, per identificare il Copper come causa dell'interrupt è stato riservato questo bit. In genere si usa quando si vuole modificare qualcosa in memoria nel momento in cui il pennello elettronico raggiunge una certa posizione.

Interrupt audio

I bit 10-7, AUD3-0, sono assegnati ai canali audio. Sono chiamati AUD3, AUD2, AUD1 e AUD0 e sono assegnati rispettivamente ai canali 3, 2, 1 e 0.

Sono tutti interrupt di livello 4, e segnalano il termine della lettura di un blocco di dati audio. Quando il DMA audio opera in modo automatico, l'interrupt si verifica non appena viene letta l'ultima word del blocco. In modo manuale, si verifica quando il registro dei dati audio è pronto ad accettare un'altra word di dati.

Su questo argomento si veda anche il Capitolo 5.

Interrupt del Blitter

Il bit 6, BLIT, segnala il temine di un blit. Se questo bit è a 1, indica che il Blitter ha terminato le sue operazioni ed è pronto per un nuovo compito. Questo bit dà origine a un interrupt di livello 3.

Interrupt dei dischi

I bit 12 e 1 sono assegnati agli interrupt riguardanti il sistema dei dischi. Il bit 12, DSKSYN, indica che il registro di sincronizzazione corrisponde ai dati letti (genera un interrupt di livello 5). Il bit 1, DSKBLK, indica il termine del trasferimento di un blocco di dati. Viene usato per indicare che una specifica richiesta di DMA è stata completata (genera un interrupt di livello 1).

Per approfondire l'argomento si veda il Capitolo 8.

Interrupt della porta seriale

I bit associati agli interrupt della porta seriale sono il bit 11 e il bit 0. Il bit 11, RBF (receive buffer full, buffer di ricezione pieno) indica il riempimento del buffer di input dell'UART (genera un interrupt di livello 5). Il bit 0, TBE (transmit buffer empty, buffer di trasmissione vuoto) indica che il buffer di output dell'UART può essere riempito con nuovi dati (genera un interrupt di livello 1).

Priorità hardware	Priorità software dell'Exec		Nome simbolico
		Descrizione	
	1	Interrupt software	SOFTINT
1	2	Blocco del disco trasferito	DSKBLK
	3	Buffer di trasmissione vuoto	TBE
2	4	Porte di I/O, CIAA e linea INT2	PORTS
	5	Copper attivo	COPER
3	6	Intervallo di vertical blanking	VERTB
	7	Blit completato	BLIT
	8	Canale audio 2	AUD2
4	9	Canale audio 0	AUD0
	10	Canale audio 3	AUD3
	11	Canale audio 1	AUD1
5	12	Buffer di ricezione pieno	RBF
	13	Sincronizzazione del disco	DSKSYNC
6	14	CIAB e linea INT6	EXTER
	15	Abilita tutti gli interrupt	INTEN
7	–	Interrupt non mascherabile	NMI

Figura 7.4: Priorità degli interrupt

Controllo dei canali DMA

Durante le operazioni di sistema si verificano molti casi di accesso diretto alla memoria. A questo scopo vi sono un indirizzo in scrittura e un indirizzo in lettura che individuano il registro di controllo del DMA: tramite questo registro si possono abilitare o disabilitare i vari canali.

DMACONR è il nome dell'indirizzo in lettura. DMACON quello dell'indirizzo in scrittura. Il significato dei loro bit è riportato nella seguente tavola.

Tavola 7.6: Contenuto dei registri di controllo del DMA

Bit	**Nome**	**Funzione**
15	SET/CLR	Bit d'impostazione/azzeramento. Si veda il precedente paragrafo sugli interrupt.
14	BBUSY	Bit di Blitter occupato - *a sola lettura*.
13	BZERO	Flag di zero del Blitter - *a sola lettura*. Rimane a 1 se durante un'operazione del Blitter l'output è stato sempre zero.
12-11		Non assegnati.
10	BLTPRI	Quando è a 1, il Blitter ha priorità completa, anziché parziale, sul 68000.
9	DMAEN	Abilitazione del DMA. Quando è a zero, risultano disabilitati tutti i canali DMA.
8	BPLEN	Abilita il DMA dei bitplane.
7	COPEN	Abilita il DMA del Copper.
6	BLTEN	Abilita il DMA del Blitter.
5	SPREN	Abilita il DMA degli sprite.
4	DSKEN	Abilita il DMA dei dischi.
3-0	AUDxEN	Abilita il DMA dei canali audio 3-0 (x = 3-0).

Per ulteriori informazioni sull'uso del DMA si consultino:

Copper Capitolo 2.
Bitplane Capitolo 3.
Sprite Capitolo 4.
Audio Capitolo 5.
Blitter Capitolo 6.
Dischi Capitolo 8.

Accesso del microprocessore alla memoria chip

I chip dell'Amiga accedono direttamente alla memoria, anziché utilizzare i tradizionali meccanismi di arbitraggio del bus. Pertanto, alcune caratteristiche del 68000 per il supporto di sistemi multiprocessore (come l'istruzione TAS) non sono previste dall'architettura dell'Amiga.

Il reset e l'inizializzazione del sistema

Quando l'Amiga viene acceso o quando viene impartito un reset, la mappa di memoria viene a trovarsi in uno stato particolare. All'indirizzo $00000000 appare una copia addizionale della ROM di sistema, e la RAM che normalmente si trova a questo indirizzo non è disponibile. In alcuni modelli vi sono zone di RAM che restano disponibili, in altri nessuna. Il software deve pertanto ritenere che non vi sia memoria disponibile. Il bit OVL in uno dei chip 8520 disabilita questa sovrapposizione (la posizione di questo bit è riportata nell'Appendice F).

La ROM contiene un codice d'identificazione nella prima word. Attenzione: in futuro il valore di questo codice potrebbe cambiare. La seconda word contiene un'istruzione JMP ($4EF9). Le due successive vengono usate come program counter iniziale dal 68000.

L'istruzione RESET del 68000 agisce in maniera molto simile a un reset esterno o all'accensione iniziale. Tutta la memoria e le schede AUTOCONFIG scompaiono, e la ROM appare all'indirizzo $00000000. La differenza è che la CPU continua con le istruzioni successive. Dal momento che la RAM potrebbe non essere più disponibile, è necessaria una particolare attenzione per scrivere codice di boot che funzioni correttamente su tutti i modelli dell'Amiga. Il seguente listato costituisce *l'unico* codice ufficialmente approvato e garantito dalla Commodore:

```
        CNOP    0,4  ;IMPORTANTE: dev'essere allineato alle long word
MagicResetCode:
        lea.l   2,a0 ;punta all'istruzione JMP all'inizio della ROM
        RESET        ;Tutta la RAM scompare!
        jmp     (a0) ;Sfrutta il prefetch per eseguire questa istruzione
```

L'istruzione RESET dev'essere eseguita mentre la CPU è in modo supervisore. Utilizzando l'Exec, si ricorra al seguente codice:

```
_ColdReboot:
        move.l  4,a6                  ;SysBase
        lea.l   MagicResetCode(pc),a5 ;Indirizzo del codice da eseguire
        jsr     _LVOSupervisor(a6)    ;Entra in modo supervisore
```

Introduzione

Questo capitolo si occupa dell'hardware attraverso il quale l'Amiga dialoga con il mondo esterno, e riguarda i seguenti dispositivi:

- Due porte di controllo riconfigurabili, destinate al mouse, al joystick, alla penna ottica.
- Controller del sistema dei dischi, per i floppy disk e per altri dispositivi MFM o GCR.
- Tastiera.
- Interfaccia parallela compatibile Centronics.
- Interfaccia seriale compatibile RS232-C, per il modem e altri dispositivi seriali.
- Connettori video: RGB, monocromatico, modulatore RF, slot video.

Interfaccia delle porte di controllo

Ogni Amiga possiede due connettori a 9 pin che possono essere usati per input/output con molti tipi di controlli. La figura mostra una delle due porte e il tipico connettore che vi viene inserito.

Figura 8.1: Porta di controllo e connessione all'Amiga

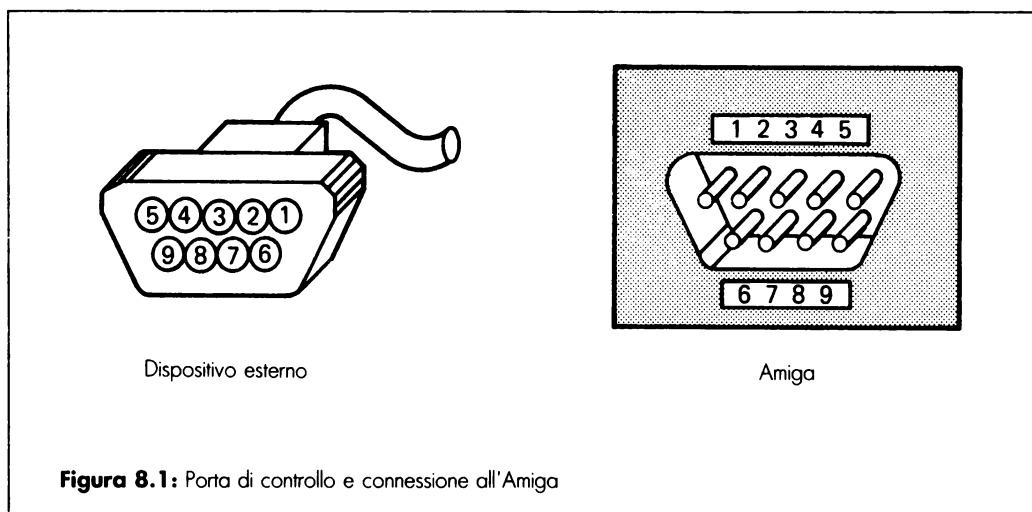



Tavola 8.1: Le più comuni connessioni dei dispositivi di controllo

Pin	Joystick	Mouse, trackball	Controllo proporzionale	Joystick proporz. X-Y	Penna ottica
1	Avanti	V	-	Pulsante 3**	-
2	Indietro	H	-	-	-
3	Sinistra	VQ	Pulsante sin.	Pulsante 1	-
4	Destra	HQ	Pulsante des.	Pulsante 2	-
5*	-	Pulsante centr.**	POT destro	POT X	Penna premuta sullo schermo
6*	Pulsante 1	Pulsante sin.	-	-	Aggancio del pennello elettronico
7	-	+5V	+5V	+5V	+5V
8	GND	GND	GND	GND	GND
9*	Pulsante 2**	Pulsante des.	POT sin.	POT Y	Pulsante 2**

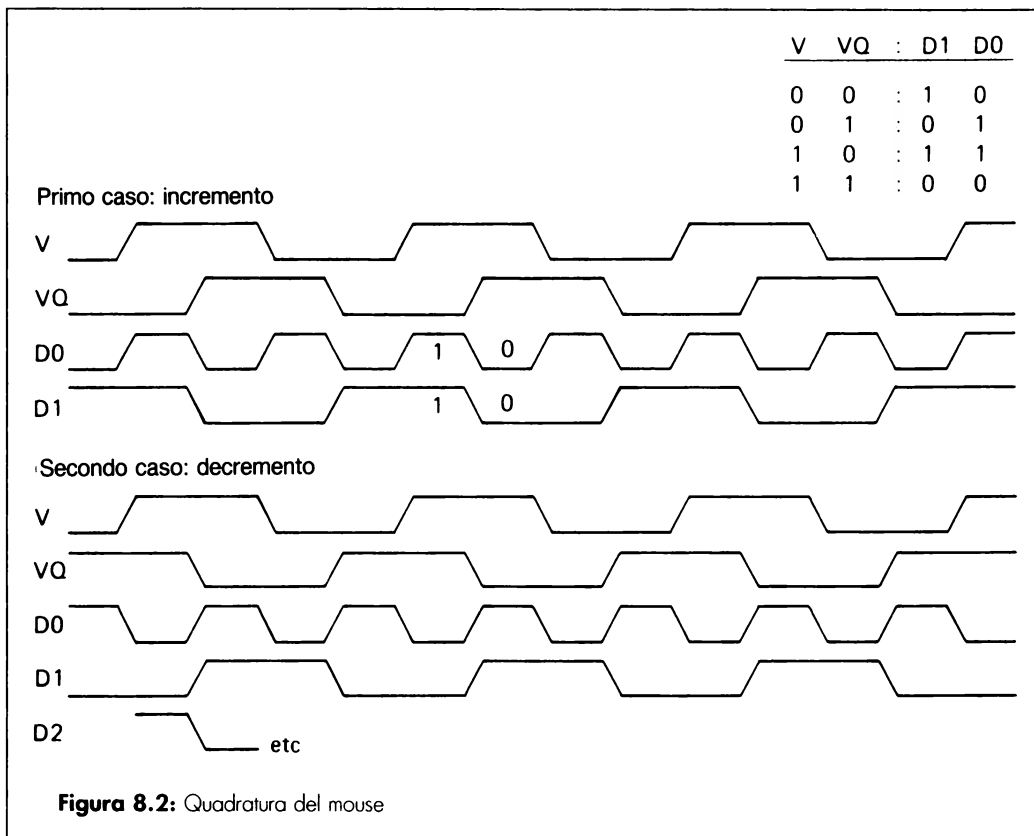
* Questi pin possono anche essere configurati come output.
** Questi collegamenti sono opzionali.

REGISTRI IMPIEGATI DALLE PORTE DI CONTROLLO

JOY0DAT	$DFF00A	Contatore per input digitale (mouse) (porta 1).
JOY1DAT	$DFF00C	Contatore per input digitale (mouse) (porta 2).
CIAAPRA	$BFE001	Input e output del pin 6 (pulsanti Fire delle porte 1 e 2).
POT0DAT	$DFF012	Contatore per input proporzionale (porta 1).
POT1DAT	$DFF014	Contatore per input proporzionale (porta 2).
POTGO	$DFF034	Inizializzazione dei contatori proporzionali.
POTGOR	$DFF016	Come sopra, ma in lettura.
BPLCON0	$DFF100	Il bit 3 abilita la penna ottica.
VPOSR	$DFF004	Posizione della penna ottica (bit più significativi).
VHPOSR	$DFF006	Posizione della penna ottica (bit meno significativi).

LETTURA DI CONTROLLI DI TIPO MOUSE O TRACKBALL

Gli impulsi in ingresso dai pin dedicati al mouse vengono convertiti in una coppia di valori (orizzontale e verticale). I registri di conteggio possono seguire i movimenti del mouse senza l'intervento della CPU.

Figura 8.2: Quadratura del mouse

Il mouse utilizza alcuni input di quadratura. Per ogni direzione, una ruota all'interno del mouse produce due serie di impulsi, sfasati di 90 gradi (si veda la Figura 8.2). La relazione fra le due fasi determina la direzione.

I contatori vengono incrementati muovendo il mouse a destra o verso il basso. Vengono decrementati muovendo il mouse a sinistra o verso l'alto.

Lettura dei contatori

Il contenuto dei registri di posizione del mouse può essere letto tramite i registri che si trovano agli indirizzi JOY0DAT e JOY1DAT, dedicati rispettivamente ai valori relativi alle porte 1 e 2.

Il contenuto di questi registri è il seguente:

Bit 15-8	Coordinata verticale del mouse
Bit 7-0	Coordinata orizzontale del mouse

Limitazioni dei contatori

Questi contatori non hanno un limite inferiore o superiore: il conteggio arriva al massimo e ricomincia da zero, o viceversa. Volendo usare il mouse per controllare qualcosa sullo schermo, si devono leggere i contatori almeno a ogni intervallo di vertical blanking ed è necessario conservare il precedente contenuto del registro. Quindi, facendo la differenza dei due valori, si possono ricavare la direzione e la velocità del movimento.

Il mouse dà origine a circa 79 incrementi o decrementi del contatore per uno spostamento di un centimetro. L'intervallo di vertical blanking si verifica una volta ogni cinquantesimo di secondo. Leggendo i registri a ogni intervallo, è molto probabile che lo spostamento sia inferiore a 127. Solo se il mouse viene mosso a una velocità maggiore di 95 cm/sec possono esserci dei problemi nel calcolo del valore corretto. In un videogioco che richieda un'azione molto veloce, potrebbe presentarsi la necessità di leggere questi registri due volte per quadro video.

Una volta sottratto il conteggio precedente da quello attuale, il valore assoluto della differenza rappresenta la velocità. Il suo segno rappresenta invece la direzione, o meglio il verso, del movimento.

Per calcolare la velocità del mouse, il modo più semplice è ricorrere all'aritmetica a 8 bit con segno. Il nuovo valore del contatore meno quello precedente rappresenta il numero di incrementi. Nella Tavola 8.2 viene comunque illustrato un metodo alternativo: entrambi i valori vengono trattati come numeri senza segno da 0 a 255. Il conteggio è di 100 impulsi in entrambi i casi.

Tavola 8.2: Come determinare lo spostamento del mouse

Conteggio precedente	Conteggio attuale	Direzione
200	100	Alto (sinistra)
100	200	Basso (destra)
200	45	Basso*
45	200	Alto**

Note per la Tavola 8.2:

* Poiché 45 - 200 = -155, che in valore assoluto è maggiore di 127, il conteggio effettivo dev'essere 255 + (45 - 200) = 100; la direzione è verso il basso.

** Poiché 200 - 45 = 155, che è maggiore di 127, il conteggio effettivo dev'essere 255 - (200 - 45) = 100; la direzione è verso l'alto.

Pulsanti del mouse

Vi sono due pulsanti sul mouse standard dell'Amiga. L'hardware tuttavia prevede fino a tre pulsanti.

- Il pulsante 1 (sinistro) è connesso al registro CIAAPRA ($BFE001). Il pulsante della porta 1 è connesso al bit 6, quello della porta 2 al bit 7. Si consulti l'Appendice F per ulteriori informazioni sugli 8520. Uno stato logico di 1 significa "circuito aperto". Uno 0 significa "circuito chiuso".

- Il pulsante 2 (destro) è connesso al pin 9 della porta di controllo, uno dei pin proporzionali. Consultare la sezione "INPUT/OUTPUT DIGITALE SULLA PORTA DI CONTROLLO" per i dettagli.

- Il pulsante 3, quando è usato, è connesso al pin 5, l'altro pin di input proporzionale.

LETTURA DI UN JOYSTICK DIGITALE

I joystick digitali contengono quattro switch direzionali, sui quali agisce la levetta di controllo. Quando la levetta viene premuta diagonalmente si attivano due switch contemporaneamente; il numero delle possibili direzioni è pertanto 8. Tutti i joystick digitali hanno almeno un pulsante di fuoco (pulsante Fire).

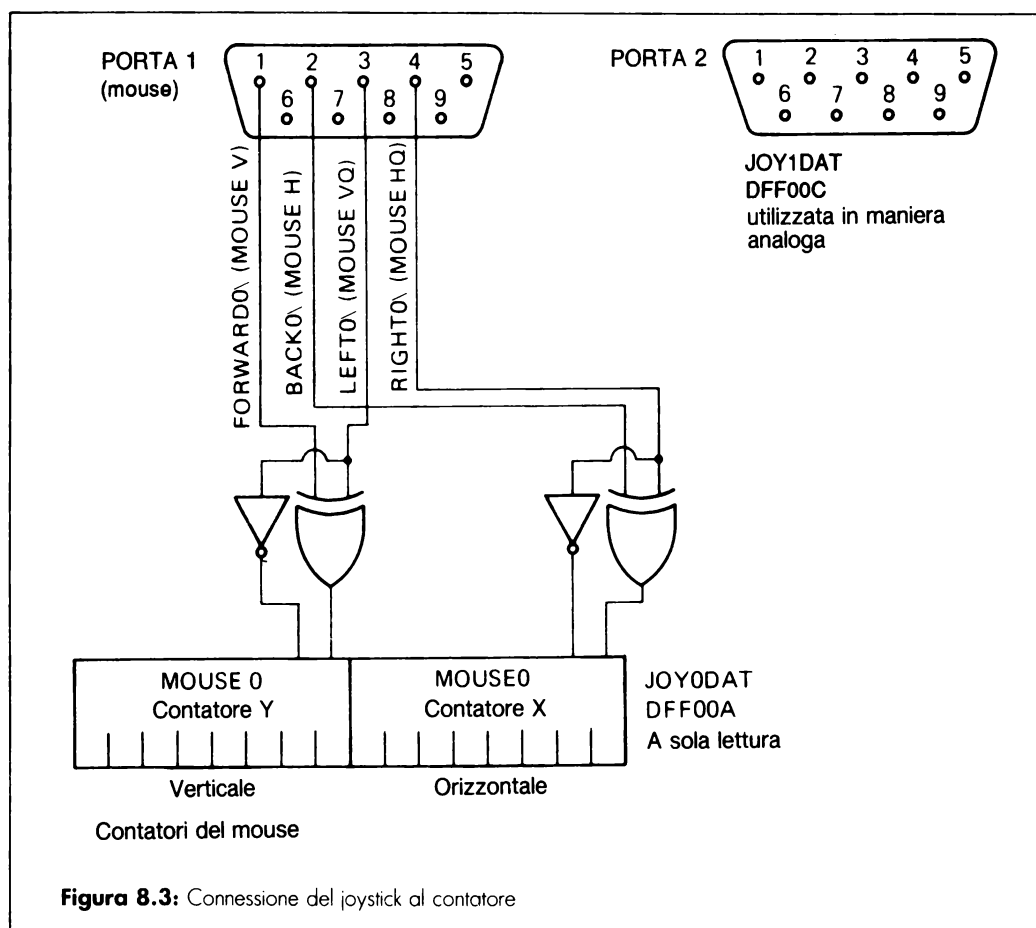
In genere gli switch sono di tipo aperto. Quando lo switch viene premuto, la corrispondente linea di input viene messa a terra. Il circuito aperto corrisponde a un 1, il circuito chiuso a uno 0.

La lettura dei dati riguardante il joystick, tuttavia, non è così semplice, poiché i registri dati sono sempre quelli usati nella gestione del mouse, JOY0DAT e JOY1DAT, come si vede nella Figura 8.3.

La Tavola 8.3 mostra come si interpretano i dati dei registri. In questi registri è 1 = circuito chiuso.

Tavola 8.3: Interpretazione di JOY0DAT e JOY1DAT

Bit di dati	Interpretazione
1	Stato logico della direzione "destra"
9	Stato logico della direzione "sinistra"
1 XOR 0	Stato logico della direzione "indietro"
9 XOR 8	Stato logico della direzione "avanti"

Figura 8.3: Connessione del joystick al contatore

I pulsanti Fire delle porte 1 e 2 sono connessi ai bit 6 e 7 del registro CIAAPRA ($BFE001). Qui uno 0 indica circuito chiuso.

Alcuni joystick hanno anche un secondo pulsante. Il suo uso è incoraggiato purché la sua funzione venga duplicata tramite la tastiera o un altro meccanismo. Viene letto esattamente come il pulsante destro del mouse.

LETTURA DEI CONTROLLI PROPORZIONALI

Ognuna delle due porte di controllo prevede l'utilizzazione di due dispositivi di input a resistenza variabile. Vi sono principalmente due tipi di controlli proporzionali: la coppia di controlli "paddle" e i joystick proporzionali X-Y. Una coppia di controlli paddle consiste in due apparecchi separati, ognuno munito di una resistenza variabile e di un pulsante Fire, collegati alla medesima porta di controllo. La connessione tipica è illustrata dalla Figura 8.4.

In un joystick proporzionale X-Y, ci sono due resistenze variabili collegate individualmente agli assi X e Y di una levetta di controllo.

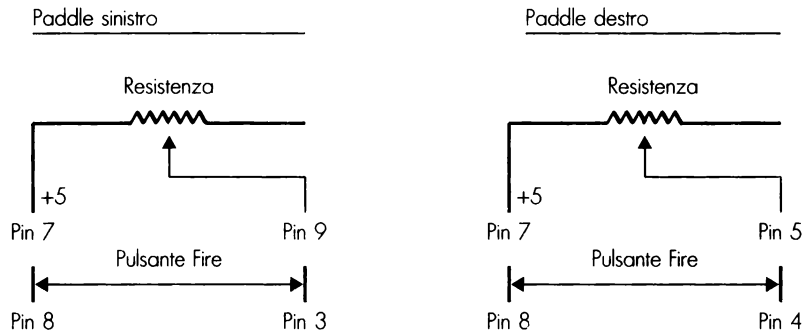



Figura 8.4: Tipico schema di collegamento di una coppia di paddle

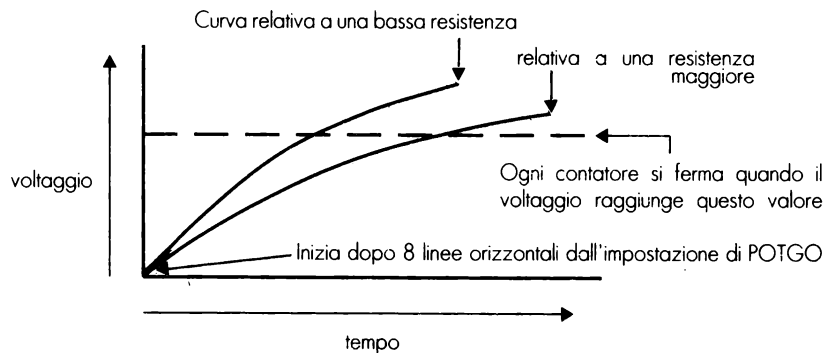



Figura 8.5: Effetti della resistenza sulla velocità di caricamento

Dispositivi proporzionali: lettura dei pulsanti

Per i controlli di tipo paddle, le direzioni destra e sinistra di un normale joystick vengono usate come pulsanti Fire per il paddle destro e quello sinistro.

Interpretazione della posizione

Interpretare la posizione tramite controlli proporzionali in genere richiede una certa quantità di lavoro durante l'intervallo di vertical blanking.

Durante questo intervallo, infatti, si deve inserire un valore nel registro POTGO. Per un normale joystick X-Y si tratta del valore $0001. Scrivere in quel registro fa sì che l'hardware inizi la lettura dei valori del potenziometro e inizializzi a zero i registri POT (si veda più avanti).

I circuiti di lettura entrano in uno stato di reset che dura per sette od otto linee di scansione. Quindi permettono la formazione di una carica su un condensatore, la cui velocità di caricamento è determinata dalla resistenza presente nel controllo esterno. Per ognuna delle linee di scansione successive, il circuito confronta la carica del condensatore con un valore prestabilito. Se la carica è ancora al di sotto di questo valore, il contatore POT viene incrementato e il confronto viene ripetuto alla linea successiva. Quando la carica supera il valore, il contatore viene "congelato" fino alla successiva scrittura del registro POTGO.

In genere POTGO viene scritto all'inizio dello schermo. Poi, nel successivo intervallo di vertical blanking viene letto il valore del registro POT, e subito dopo POTGO viene nuovamente riscritto, e così via.

Non vi è nulla nel sistema che impedisca al contatore di superare il valore massimo (255) e ricominciare da zero. Tuttavia, l'hardware è progettato in modo da garantire che ciò non avvenga nello spazio di un solo quadro video. Pertanto è sempre possibile sapere con sicurezza quando si verifica un overflow.

Registri dei controlli proporzionali

I controlli proporzionali utilizzano i seguenti registri:

POT0DAT	Dati della porta 1 (orizzontale e verticale)
POT1DAT	Dati della porta 2 (orizzontale e verticale)

Significato dei bit:

Bit 15-8 POT0Y o POT1Y
Bit 7-0 POT0X o POT1X

Tutti questi valori vengono riportati a zero quando si accede in scrittura al registro POTGO, impostandone a 1 il bit 0. I valori in genere vengono letti durante il quadro video successivo all'abilitazione.

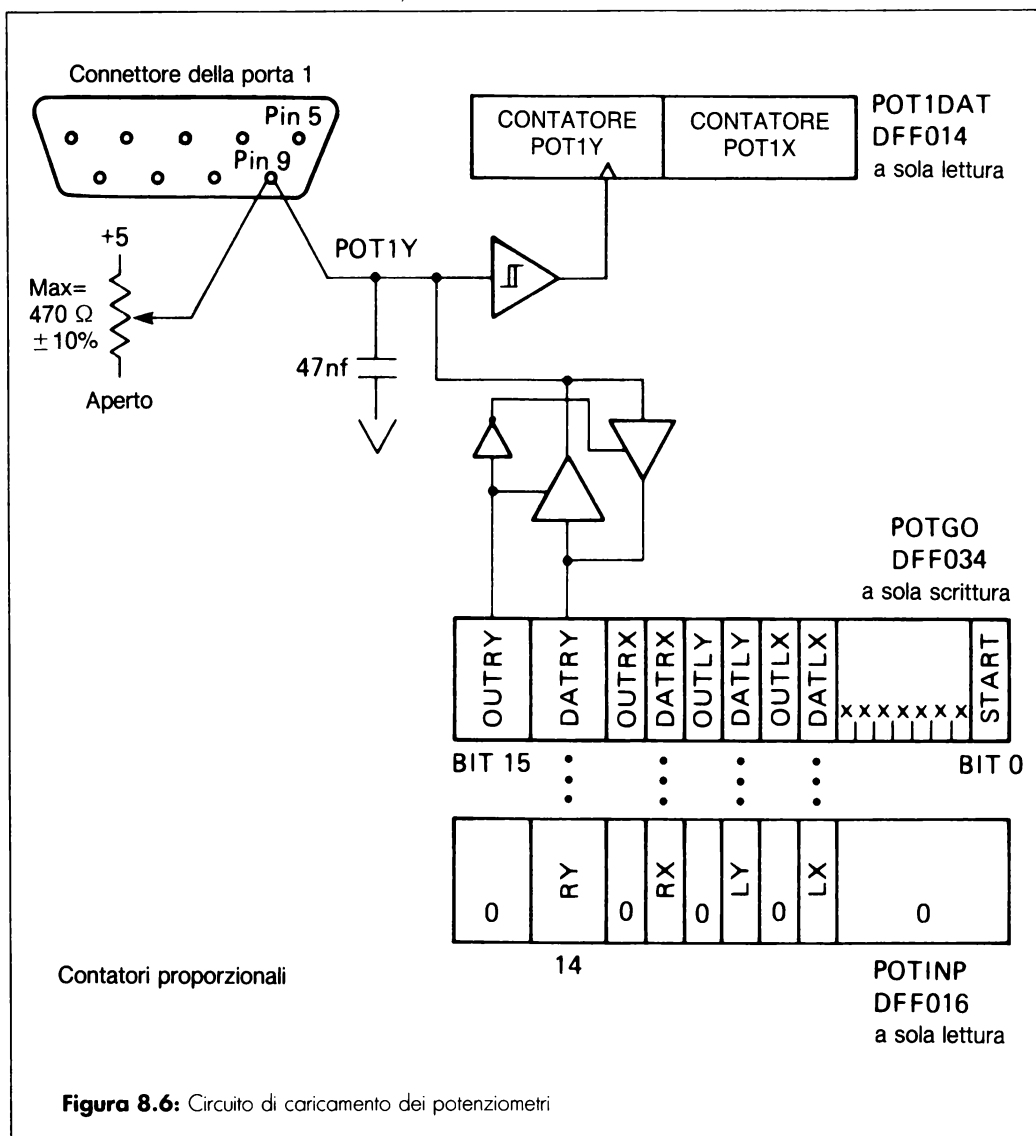
Specifiche del potenziometro

La resistenza del potenziometro deve crescere linearmente. A causa della tecnologia del convertitore analogico-digitale, la resistenza massima non deve superare i 528 Ω (sono suggeriti 470 Ω +/- 10%) per i potenziometri X e Y. Il sistema è basato su un condensatore di capacità 0,047 μF, +/- 10%, e un tempo massimo di circa 20 millisecondi (un quadro video) per raggiungere la piena carica.

Tutti i potenziometri soffrono di una certa imprecisione. Per ottenere risultati accettabili con un'ampia gamma di possibili configurazioni, si deve ricorrere alla media di più letture successive.

LA PENNA OTTICA

Alle porte di controllo si può collegare anche una penna ottica. Sull'Amiga 1000 la penna dev'essere collegata alla porta 1. Sugli Amiga 500 e 2000 la porta di default è la 2, ma si può anche selezionare la porta 1 agendo su un jumper interno. Questa modifica richiede comunque pochissime variazioni a livello di programma. Il metodo di controllo prescinde completamente

Figura 8.6: Circuito di caricamento dei potenziometri

dalla porta usata.

Il segnale chiamato "penna premuta sullo schermo" è generato da un interruttore situato nella punta della penna. Il segnale è collegato a un pin di input del potenziometro, e pertanto dev'essere letto nello stesso modo del pulsante destro del mouse o di quello centrale.

Il principio di funzionamento della penna ottica è il seguente:

1. Non appena il sistema esce dall'intervallo di vertical blanking, la circuiteria della penna viene automaticamente abilitata.

2. Il pennello elettronico inizia a creare l'immagine sullo schermo, muovendosi da sinistra a destra e dall'alto in basso.

3. Il sensore della penna avverte l'impulso luminoso e invia quindi il segnale di "aggancio del pennello elettronico".

4. Questo segnale informa l'hardware che deve salvare il contenuto del registro VPOSR (il che permette di sapere qual è la posizione esatta della penna nel momento in cui incontra il pennello elettronico).

Lettura dei registri della penna ottica

I registri riguardanti la penna ottica sono gli stessi del pennello elettronico, VPOSR e VHPOSR. I bit sono definiti come segue:

VPOSR:	Bit 15	Quadro lungo/quadro breve (0 = quadro breve).
	Bit 14-1	Non utilizzati.
	Bit 0	V8 (bit più significativo della posizione verticale).
VHPOSR:	Bit 15-8	V7-V0 (posizione verticale).
	Bit 7-0	H8-H1 (posizione orizzontale).

Da programma è possibile utilizzare questa coppia di registri come una long word il cui indirizzo sia VPOSR.
La risoluzione di posizione dei registri è la seguente:

Verticale:	1 linea di scansione in modo non interlace. 2 linee di scansione in modo interlace (sapendo se è in corso il quadro video lungo o quello breve si può determinare la posizione esatta).
Orizzontale:	2 pixel del tipo bassa risoluzione, sia in alta che in bassa risoluzione.

L'imprecisione nella misura dipende in larga parte dalla qualità della penna ottica. Nella maggior parte delle applicazioni conviene ricorrere alla media di più letture consecutive.
Per abilitare l'input dalla penna ottica occorre porre a 1 il bit 3 di BPLCON0. Fatto ciò, ogniqualvolta la penna ottica invia il segnale di aggancio del pennello elettronico, il valore contenuto in VPOSR viene "congelato" (in effetti il contatore continua a registrare; soltanto il valore in lettura viene bloccato). Questo congelamento ha termine alla fine dell'intervallo di vertical blanking (cioè nella posizione verticale 20). Non vi è un bit che stia a indicare che l'aggancio è avvenuto, ma l'evento può essere accertato come segue:

1. Si legge due volte (come long word) il registro VPOSR.

2. Se i due valori non sono uguali, la penna ottica non è stata attivata dopo l'ultimo inizio di schermo (V = 20).

3. Se i due valori sono uguali, si mascherano i 15 bit più significativi della long word e la si confronta con il valore esadecimale $13700 (V = 311).

4. Se il valore di VPOSR è maggiore di $13700, la penna ottica non è stata attivata dopo l'ultimo inizio di schermo. Se è minore, la penna ha agganciato il pennello elettronico e il valore letto è effettivamente la posizione sullo schermo della penna ottica.

Un metodo per alcuni versi più semplice per determinare la validità della lettura consiste nel far sì che il software legga il registro VPOSR *soltanto* durante il periodo di vertical blanking ($0 < V < 20$):

1. Si legge VPOSR una sola volta (come long word) nel periodo $0 < V < 20$.
2. Si mascherano i 15 bit più significativi della long word e la si confronta con il valore esadecimale $13700 (V = 311).
3. Se il valore di VPOSR è maggiore di $13700, la penna ottica non è stata attivata dopo l'ultimo inizio di schermo. Se è minore, la penna ha agganciato il pennello elettronico e il valore letto è effettivamente la posizione sullo schermo della penna ottica.

Si noti che in qualunque momento il registro VPOSR può essere bloccato, e in tal caso non può essere usato come contatore. Questo comportamento può causare problemi con i programmi che utilizzano VPOSR per le loro temporizzazioni.

INPUT/OUTPUT DIGITALE SULLA PORTA DI CONTROLLO

L'Amiga può leggere e interpretare dati provenienti da molti dispositivi di controllo non standard. Le linee di controllo presenti nel registro POTGO ($DFF034) possono ridefinire la funzione di alcuni dei pin delle porte di controllo.

La Tavola 8.4 descrive i bit del registro POTGO. POTGO ($DFF034) è l'indirizzo a sola scrittura di questo registro di controllo. POTINP ($DFF016, chiamato anche POTGOR) è l'indirizzo a sola lettura. Questo registro controlla una porta di I/O bidirezionale a 4 bit, che condivide i suoi 4 pin con i controlli proporzionali.

Tavola 8.4: I registri POTGO e POTINP

Bit	Nome	Funzione
15	OUTRY	Abilitazione dell'output per il bit 14 (output = 1)
14	DATRY	Dati per la porta 2, pin 9
13	OUTRX	Abilitazione dell'output per il bit 12
12	DATRX	Dati per la porta 2, pin 5
11	OUTLY	Abilitazione dell'output per il bit 10
10	DATLY	Dati per la porta 1, pin 9
9	OUTLX	Abilitazione dell'output per il bit 8
8	DATLX	Dati per la porta 1, pin 5
7 -1	X	Numero di revisione del chip
0	START	Avvio dei potenziometri (scarica i condensatori, azzera i contatori)

Anziché come input per dispositivi a resistenza variabile, i pin 5 e 9 delle porte possono essere usati come una porta di I/O a quattro bit. La conseguenza diretta è la disponibilità di due pin in più per generiche operazioni di I/O.

Se uno qualsiasi di questi pin viene ridefinito come output tramite il corrispondente bit di abilitazione, l'Amiga sconnette la circuiteria di controllo del potenziometro dalla porta, e il pin viene riconfigurato come output. Il bit di dati corrispondente conterrà quindi lo stato logico del

pin. Si utilizza l'indirizzo POTGO per accedere in scrittura a questo registro, e l'indirizzo POTINP per accedervi in scrittura. Attenzione: vi sono condensatori di capacità molto elevata su queste linee, e possono essere necessari fino a 300 microsecondi per un cambiamento di stato.

Per utilizzare l'intero registro come input, controllando lo stato dei pin del potenziometro, si deve inizializzare il registro POTGO con il valore $0000. Fatto ciò, lo stato dei pin può essere letto tramite il registro POTINP. Si noti che i bit definiti come input sono collegati ai contatori proporzionali (vedere la descrizione del bit START di POTGO.)

Queste linee possono anche essere usate come input per pulsanti. Un pulsante è in genere costituito da un interruttore aperto che quando è premuto viene collegato a terra. L'Amiga deve presentare una resistenza di pull-up sul pin corrispondente. Perché ciò avvenga, è sufficiente configurare il pin come output e portare la linea a 1 (impostare OUT... e DAT... a 1). Il registro POTINP conterrà il valore 0 se il pulsante è premuto, 1 se è rilasciato.

Anche il pulsante Fire può essere configurato come output. Il registro CIAADDRA ($BFE201) contiene una maschera che corrisponde esattamente al registro di lettura dei dati, CIAAPRA ($BFE001). Impostando a 1 uno dei suoi bit, lo si trasforma in un output. Per maggiori dettagli si consulti l'Appendice F.

Controller dei dischi

Il controller dei dischi può controllare fino a quattro dispositivi di tipo MFM. In genere si tratta di disk drive per dischi a doppia faccia, doppia densità da 3,5" o 5,25". Un disk drive da 3,5" è installato nell'unità centrale.

Il controller è estremamente flessibile. Può leggere tramite DMA un'intera traccia di dati MFM in una singola rivoluzione del disco. Registri speciali permettono alla CPU di sincronizzarsi con dati specifici, o di leggere i dati in input un byte per volta. Il controller può leggere e scrivere virtualmente qualsiasi disco MFM a doppia densità, incluso il formato Amiga V1.0, MS-DOS 5,25", MS-DOS 3,5" e gran parte dei formati CP/M. Il controller può inoltre gestire la maggior parte dei dischi che utilizzano il sistema GCR (Group Coded Recording), inclusi quelli per Apple II. Variando opportunamente la velocità del motore, possono essere letti e scritti dischi anche nel formato Commodore 1541/1571.

REGISTRI USATI DAL SISTEMA DEI DISCHI

Il sistema di controllo dei dischi utilizza due porte all'interno dei chip 8520 CIA e diversi registri del chip Paula:

CIAAPRA	$BFE001	4 bit di input
CIABPRB	$BFD100	8 bit di output
ADKCON	$DFF09E	Bit di controllo - scrittura
ADKCONR	$DFF010	Bit di controllo - lettura
DSKPTH	$DFF020	Puntatore per il DMA (32 bit)
DSKLEN	$DFF024	Lunghezza del DMA
DSKBYTR	$DFF01A	Byte di dati e lettura dello stato
DSKSYNC	$DFF07E	Contiene la word di sincronismo

CIAAPRA/CIABPRB

La tavola seguente riporta l'elenco dei bit dei chip 8520 usati dal sistema dei dischi. I bit denominati "PA" sono bit di input contenuti nel registro CIAAPRA ($BFE001). I bit denominati "PB" sono bit di output situati nel registro CIABPRB ($BFD100). Altre informazioni sui chip 8520 si trovano nell'Appendice F.

Tavola 8.5: Bit dei chip 8520 usati dal sistema dei dischi

Bit	Nome	Funzione
PA5	DSKRDY*	Disco pronto (attivo basso). Il disk drive invia questo segnale quando il motore ruota alla velocità corretta. Questo segnale è valido solo quando è attivo anche il segnale DSKMOTOR*.
PA4	DSKTRACK0*	Traccia 0 (attivo basso). Il drive invia questo segnale quando la testina è sulla traccia 0. In questa situazione un programma non deve mai tentare di muoverla verso l'esterno: alcuni drive rifiutano di muoversi, ma altri tentano il movimento, il che potrebbe provocare il disallineamento della testina. Tutti i nuovi disk drive devono rifiutarsi di muovere la testina verso l'esterno in questa situazione.
PA3	DSKPROT*	Disco protetto dalla scrittura (attivo basso).
PA2	DSKCHANGE*	Disco rimosso dal drive (attivo basso). Il disk drive invia questo segnale ogni volta che viene tolto un disco dal suo interno. Continua a inviare il segnale fino a che non ne viene inserito uno nuovo e non arriva un impulso DSKSTEP*.
PB7	DSKMOTOR*	Controllo del motore (attivo basso). È un segnale non standard sul sistema Amiga. Ogni disk drive controlla questo segnale quando viene selezionato. Il motore rimane nello stesso stato fino a che la linea di selezione non viene nuovamente attivata. DSKMOTOR* controlla anche l'attività del LED del disk drive. Tutto il software che seleziona i disk drive deve impostare questo segnale *prima* della selezione. Il disk drive "ricorda" lo stato del motore, quando non è selezionato. I motori di tutti i disk drive vengono disattivati dopo il reset del sistema. Dopo aver attivato il motore, il software deve attendere per 500 ms, oppure finché il drive non invia il segnale DSKRDY*.
PB6	DSKSEL3*	Selezione del drive 3 (attivo basso).
PB5	DSKSEL2*	Selezione del drive 2 (attivo basso).
PB4	DSKSEL1*	Selezione del drive 1 (attivo basso).

PB3	**DSKSEL0***	Selezione del drive 0 (drive interno) (attivo basso).
PB2	DSKSIDE	Specifica quale faccia del disco usare. Zero indica la faccia superiore. DSKSIDE dev'essere stabile per almeno 100 microsecondi prima della scrittura. Dopo la scrittura si devono lasciar passare almeno 1,3 millisecondi prima di effettuare una modifica a DSKSIDE.
PB1	DSKDIREC	Specifica in quale direzione si deve muovere la testina. Zero indica uno spostamento verso il centro del disco. La traccia zero è la traccia più esterna. Questo segnale dev'essere inviato *prima* del segnale di movimento, tramite un'operazione di scrittura separata.
PB0	DSKSTEP*	Muove la testina del drive (attivo basso). Questo segnale deve sempre essere usato come un impulso veloce (alto, momentaneamente basso, di nuovo alto). La testina dei disk drive usati dall'Amiga raggiunge la traccia successiva in un massimo di 3 millisecondi. Loop che utilizzano il decremento di un contatore per calcolare questo intervallo **non sono accettabili**. Per una soluzione corretta si veda l'Appendice F. Quando s'inverte la direzione di movimento è necessario un minimo di 18 millisecondi dall'ultimo impulso di step. Il tempo di stabilizzazione è di 15 millisecondi.
FLAG	DSKINDEX*	Bit 4 di $BFDD00. Può essere usato per creare un interrupt di livello 6. Si veda l'Appendice F per ulteriori dettagli.

Controllo del DMA

In genere i dati vengono trasferiti al disco tramite accesso diretto alla memoria (DMA). Il controllo del DMA dipende da quattro elementi:

- Un puntatore all'area dalla quale o nella quale devono essere trasferiti i dati.
- Lunghezza dei dati da trasferire.
- Direzione del trasferimento dei dati (lettura o scrittura).
- Abilitazione del DMA.

DSKPTH – Puntatore ai dati

L'inizio dell'area dati viene specificato tramite un indirizzo a 32 bit. Il bit meno significativo dev'essere azzerato e il buffer deve trovarsi nella memoria di tipo chip. L'indirizzo dev'essere scritto come long word nel registro DSKPTH ($DFF020).

DSKLEN – Lunghezza, direzione, abilitazione del DMA

Tutti questi bit di controllo sono contenuti nel registro a sola scrittura DSKLEN:

Tavola 8.6: Il registro DSKLEN

Bit	Nome	Funzione
15	DMAEN	Abilitazione secondaria del DMA dei dischi
14	WRITE	Direzione (RAM→disco se = 1)
13-0	LENGTH	Numero di word da trasferire

L'hardware richiede una speciale sequenza di operazioni per iniziare il DMA. Ciò impedisce il verificarsi di scritture accidentali sul disco. In pratica, il bit DMAEN dev'essere impostato due

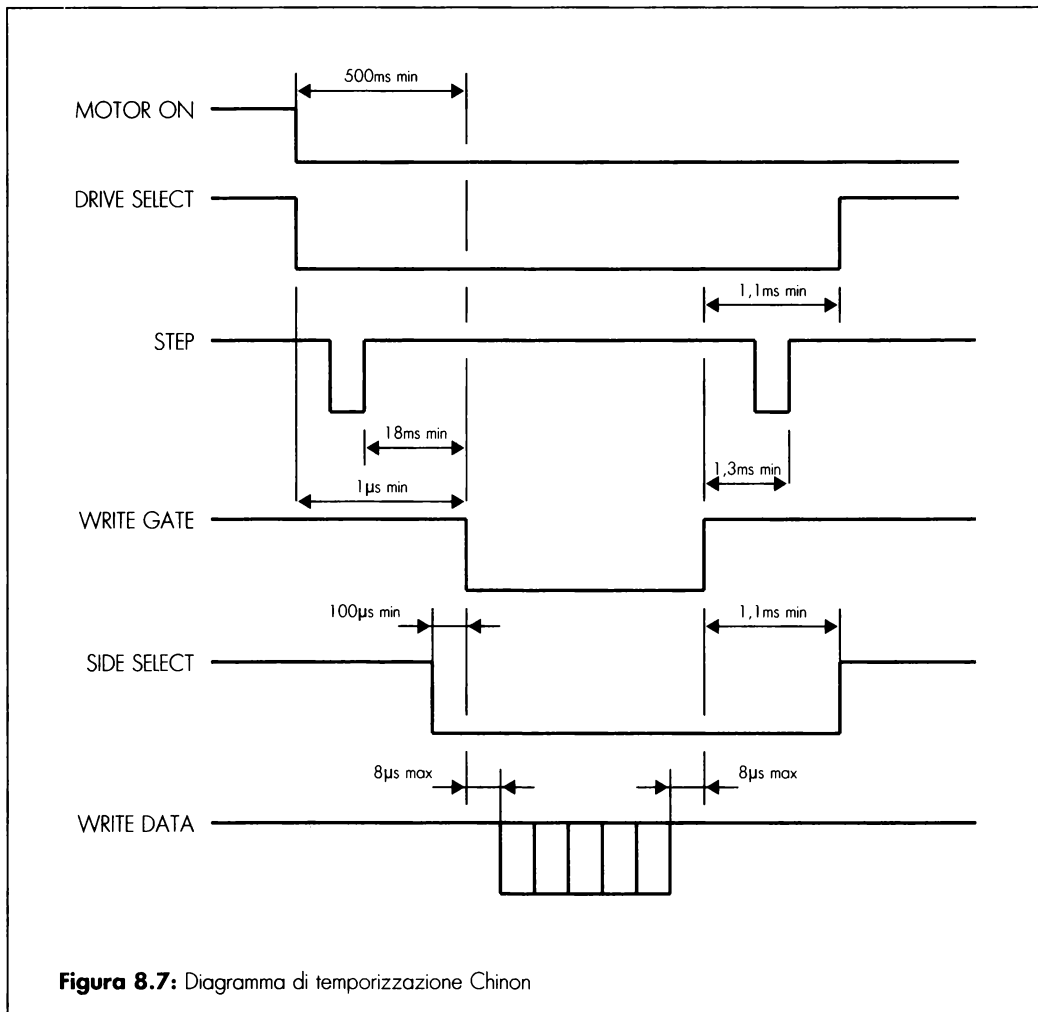



Figura 8.7: Diagramma di temporizzazione Chinon

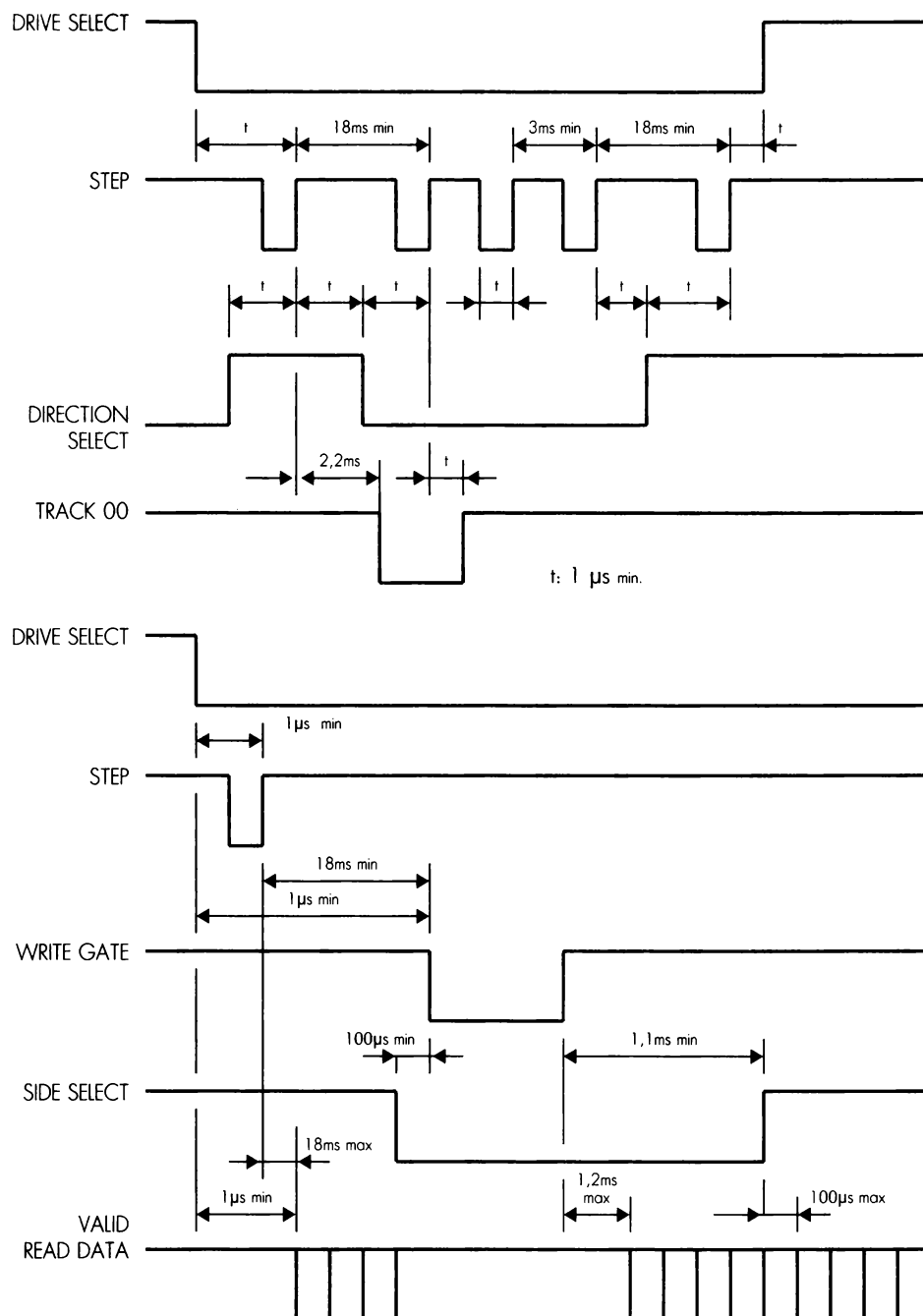



Figura 8.8: Diagramma di temporizzazione di Chinon (seconda parte)

volte per abilitare veramente il DMA. Ecco la sequenza da eseguire:

1. Si abilita il DMA nel registro DMACON (si veda anche il Capitolo 7).
2. Si scrive il valore $4000 nel registro DSKLEN, disabilitando esplicitamente il DMA.
3. S'inserisce il valore desiderato nel registro DSKLEN.
4. Lo si inserisce nuovamente (questo dà effettivamente il via al DMA).
5. Quando il DMA è stato completato, si reinserisce il valore $4000 nel registro DSKLEN, per prevenire accessi accidentali al disco.

Quando una word di dati viene trasferita, il contatore di lunghezza si decrementa e il puntatore ai dati passa alla word successiva. Quando il contatore di lunghezza raggiunge lo zero, il trasferimento ha termine.

Per la lettura si raccomanda di trasferire un'intera traccia in un buffer e cercare quindi il settore desiderato. L'uso del registro DSKSYNC (si veda più avanti) garantisce il corretto allineamento dei dati. In questo modo diventa necessario accedere al disco solo una volta per ogni traccia. In un caricatore ad alta velocità, lo spostamento della testina alla traccia successiva può avvenire mentre i dati precedenti vengono decodificati e controllati. In questo modo non vi sono tempi particolarmente critici nella lettura dei dati, e altri dispositivi come la grafica e l'audio possono procedere regolarmente nei loro compiti.

Se non vi è abbastanza memoria libera per utilizzare questo sistema, l'hardware offre alcune possibilità per cercare una determinata traccia: esiste un registro che si può leggere per esaminare il flusso dei dati in ingresso.

Nota: nel sistema esiste un bug hardware che causa la perdita degli ultimi tre bit di dati inviati al disco. Inoltre, in un'operazione di lettura l'ultima word può non essere letta (cioè viene letta una word di dati in meno di quanto richiesto).

DSKBYTR

Questo registro è il buffer-dati temporaneo del sistema dei dischi. Durante la lettura, i dati provenienti dal disco vi vengono inseriti un byte per volta. Il bit DSKBYT viene impostato a 1 quando un nuovo byte di dati viene ricevuto dal registro, mentre viene azzerato quando si accede in lettura al registro.

DSKBYTR può essere usato dalla CPU per effettuare la sincronizzazione con la rotazione del disco prima d'iniziare un'operazione di lettura o scrittura tramite DMA.

Tavola 8.7: Il registro DSKBYTR

Bit	Nome	Funzione
15	DSKBYT	Questo bit a 1 indica che il registro contiene un byte di dati. Viene azzerato durante la lettura del registro.

14	DMAON	Indica se il DMA è realmente abilitato. Se vale 1, significa che sono impostati tutti i bit di abilitazione, ovvero il bit DMAEN nel registro DSKLEN e i bit DMAEN e DSKEN in DMACON.
13	DISKWRITE	Il bit di scrittura (in DSKLEN) è stato impostato.
12	WORDEQUAL	Indica che il registro DSKSYNC è uguale al valore dei dati in input. Questo bit vale 1 solo quando i dati in ingresso coincidono con il valore contenuto nel registro di sincronizzazione (può durare anche solo 2 microsecondi).
11-8		Non utilizzati.
7-0	DATA	Byte di dati provenienti dal disco.

ADKCON e ADKCONR

ADKCON è l'indirizzo a sola scrittura di questo registro; ADKCONR quello a sola lettura. Non tutti i suoi bit sono dedicati al sistema dei dischi. Il bit 15 permette d'impostare o azzerare alcuni particolari bit indipendentemente dagli altri. Se in un'operazione di scrittura il bit 15 è a 1, tutti i bit a 1 nelle posizioni 0-14 causano l'impostazione del bit corrispondente. Se il bit 15 è a 0, ne provocano invece l'azzeramento.

Tavola 8.8: I registri ADKCON e ADKCONR

Bit	**Nome**	**Funzione**
15	SET/CLR	Permette d'impostare o azzerare i singoli bit senza modificare il resto del registro. Se il bit 15 è a 1, i bit specificati vengono impostati. Se il bit 15 è a 0, i bit specificati vengono azzerati.
14-13	PRECOMP1-0	Bit di precompensazione. Il valore 00 indica nessuna precompensazione. Il valore 01 indica 140 ns. Il valore 10 indica 280 ns. Il valore 11 indica 560 ns.
12	MFMPREC	Il valore 0 indica precompensazione di tipo GCR. Il valore 1 indica precompensazione di tipo MFM.
10	WORDSYNC	Il valore 1 abilita la sincronizzazione dell'inizio del DMA con la lettura della word contenuta nel registro DSKSYNC ($DFF07E).
9	MSBSYNC	Il valore 1 abilita la sincronizzazione con il bit più significativo in input (usato in genere in GCR).

8	FAST	Il valore 1 seleziona due microsecondi per bit (in genere MFM). I dati devono essere MFM. Il valore 0 seleziona quattro microsecondi per bit (in genere GCR).

I dati MFM da sottoporre al controller hanno dimensioni doppie dei dati non codificati. La seguente tabella mostra quali sono i rapporti:

1→01
0→10 (se il valore precedente è uno 0)
0→00 (se il valore precedente è un 1)

Per esempio:
178 = $B2 = %1011 0010
in MFM diventa
%0100 0101 0010 0100 = $4524

Con un'accorta gestione, il Blitter può essere usato come codificatore/decodificatore dei dati.
Una forma comune di codifica GCR prevede che ogni byte di dati abbia il bit più significativo impostato a 1. Quando MSBSYNC è impostato a 1, il controller cerca questo bit di sincronizzazione in ogni byte di dati. Leggendo un disco formattato in GCR, i programmi devono usare un'apposita tabella di traduzione che assicuri che i dati non contengano troppe serie consecutive di 0 o di 1.

DSKSYNC

Il registro DSKSYNC è usato per sincronizzare il flusso di dati in ingresso, una capacità particolarmente utile nella lettura dei dischi. Se il bit WORDSYNC del registro ADKCON è a 1, nessun dato viene trasferito finché nei dati in ingresso non viene trovata una word corrispondente al valore contenuto in questo registro. Il trasferimento tramite DMA ha inizio dalla word successiva. In genere il valore contenuto in DSKSYNC è il "magico" valore MFM di sincronizzazione, $4489.
Quando i dati in ingresso corrispondono al contenuto del registro DSKSYNC viene impostato anche il bit DSKSYNC del registro INTREQ. Questa impostazione non dipende in alcun modo dal bit WORDSYNC.

INTERRUPT DEI DISCHI

Il controller dei dischi può dare origine a tre tipi di interrupt:

- DSKSYNC (livello 5, bit 12 di INTREQ) – quando i dati in ingresso corrispondono al contenuto del registro DSKSYNC.
- DSKBLK (livello 1, bit 1 di INTREQ) – la richiesta di DMA è stata completata.
- INDEX (livello 6, pin FLAG dell'8520) – è scattato il sensore di indice.

Per ulteriori informazioni sull'argomento, si veda la descrizione del registro DSKLEN e l'Appendice F.

La tastiera

La tastiera è collegata al sistema tramite il registro di shift seriale contenuto in uno dei chip 8520. La linea dati è collegata al pin SP, il clock della tastiera al pin CNT. L'Appendice H contiene una descrizione completa di questa interfaccia.

COME VENGONO RICEVUTI I DATI PROVENIENTI DALLA TASTIERA

La linea CNT viene usata come clock per la tastiera. A ogni transizione di questa linea, un bit di dati giunge dalla tastiera. Il segnale di clock viene trasmesso dalla tastiera quando vi è un bit disponibile sulla linea SP. Si tratta di un impulso attivo basso.

Quando ha ricevuto dalla tastiera un byte completo, il chip 8520 genera un interrupt della CPU. La tastiera resta in attesa di un segnale di conferma da parte del sistema prima di trasmettere altri dati. Il segnale consiste nel portare la linea SP prima bassa e poi alta. Anche se alcuni modelli di tastiera possono reagire a un impulso di un solo microsecondo, per funzionare correttamente con tutti i modelli di tastiera il segnale deve durare almeno 85 microsecondi.

Se un altro tasto viene premuto prima che il precedente sia stato accettato dal sistema, il microprocessore della tastiera ne memorizza il codice in un apposito buffer che può contenere un massimo di 10 codici.

TIPO DEI DATI

I dati provenienti dalla tastiera *non* vengono ricevuti sotto forma di caratteri ASCII. Per garantire una maggior versatilità, vengono infatti trasmessi sotto forma di codici. Questi codici includono sia la pressione che il rilascio dei tasti. Ciò permette al software di sapere con esattezza quello che accade sulla tastiera.

Segue una lista dei valori esadecimali assegnati alla tastiera. La pressione di un tasto dà origine al valore corrispondente. Il rilascio dà invece origine al medesimo valore più $80.

Si noti che i codici forniscono unicamente un'informazione di posizione: i simboli corrispondenti ai vari tasti possono variare a seconda della tastiera internazionale usata.

CODICI: $00-$3F

Questi sono i codici corrispondenti al corpo principale della tastiera. Le lettere relative ai vari tasti cambiano per i diversi Paesi; non tutti i Paesi utilizzano tastiere di tipo QWERTY. Conviene quindi illustrare i codici a seconda delle posizioni indicate nelle figure 8.9 e 8.10. Alcune tastiere internazionali hanno due tasti in più, ricavati dal rimpicciolimento di tasti più grossi. Corrispondono ai codici $30 (ricavato dallo Shift sinistro) e $2B, (ricavato dal tasto Return).

CODICI: $40-$5F (comuni a tutte le tastiere)

40	Spazio
41	Backspace
42	Tab
43	Tasto "ENTER" della tastierina numerica
44	Return

45	Escape
46	Delete
4C	Freccia verso l'alto
4D	Freccia verso il basso
4E	Freccia verso destra
4F	Freccia verso sinistra
50-59	Tasti funzione F1-F10
5F	Help

CODICI: $60-$67 (codici dei tasti qualificatori)

60	Shift sinistro
61	Shift destro
62	Caps Lock
63	Control
64	Alt sinistro
65	Alt destro
66	Amiga sinistro (o tasto Commodore)
67	Amiga destro

CODICI: $F0-$FF

Questi codici vengono usati per la comunicazione con il 68000 e non sono associati a nessun tasto. Non vi è pressione o rilascio, perciò sono completamente definiti da codici a 8 bit:

78	Avvertimento di reset. Ctrl-Amiga-Amiga è stato premuto. La tastiera attende un massimo di dieci secondi (a seconda dei vari modelli) prima di procedere al reset della macchina.
F9	L'ultimo codice trasmesso era errato. Il prossimo è quello corretto ritrasmesso.
FA	Overflow del buffer di tastiera.
FC	L'autodiagnosi della tastiera ha avuto esito negativo. Inoltre, il LED del tasto CapsLock lampeggia per indicare la causa dell'errore. Lampeggia una volta se l'errore è causato dalla ROM, due volte se è causato dalla RAM, tre volte se non funziona il timer di guardia.
FD	Inizio della sequenza di avvio (per tasti premuti al momento dell'accensione).
FE	Termine della sequenza di avvio.

Questi eventi in genere vengono filtrati dal programma di gestione della tastiera.

LIMITAZIONI DELLA TASTIERA

La tastiera dell'Amiga è formata da una matrice di righe e di colonne con un tasto a ogni intersezione (si veda l'Appendice H). Può quindi accadere che vengano generati dei caratteri "fantasma". Mentre ciò non pone problemi nella normale battitura dei testi, potrebbero esserci complicazioni con alcuni giochi che richiedono la pressione simultanea di più tasti. Esaminando la matrice si osserva quali tasti possono interferire tra loro e quali sono sempre sicuri.

I tasti fantasma compaiono quando vengono premute simultaneamente certe combinazioni di tasti. Per esempio, se tenendo premute la "A" e la "S" si preme di seguito la "Z" sulla tastiera originale dell'Amiga 1000 vengono generate la "Z" e una "X" fantasma. A partire dall'Amiga 500,

la tastiera è in grado di riconoscere semplici situazioni come quella descritta: invece di trasmettere un carattere fantasma, il controller sospende la trasmissione di qualsiasi carattere fino a che la matrice non è stata ripulita (rilasciando il tasto "A" o "S"). Alcune tastiere di alto livello possono interpretare qualunque combinazione simultanea di tasti.

Tutte le tastiere sono disegnate in maniera tale da non generare caratteri fantasma durante l'uso normale, ma soltanto quando vengono premute insolite combinazioni di tasti come quella ora citata. Di solito l'utente medio non ha abbastanza dita per generare un carattere fantasma.

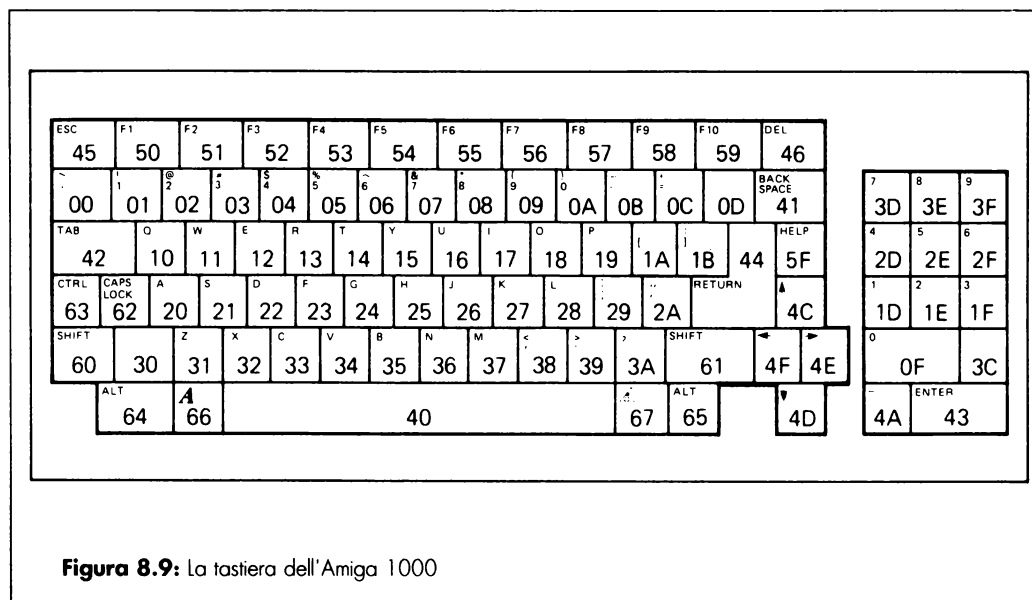



Figura 8.9: La tastiera dell'Amiga 1000

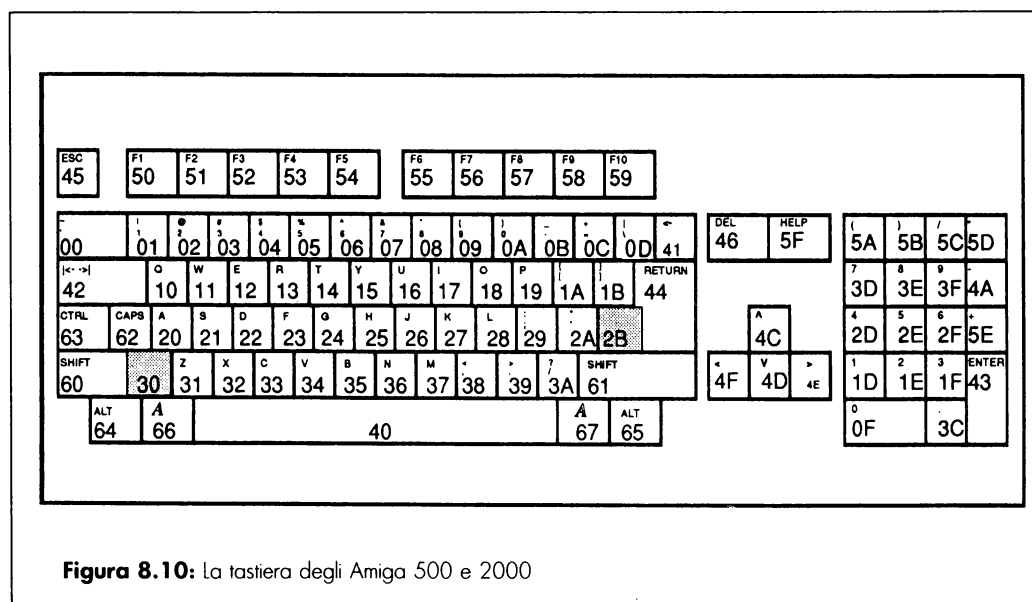



Figura 8.10: La tastiera degli Amiga 500 e 2000

Sette tasti non fanno parte della matrice e non contribuiscono in nessun caso alla generazione di caratteri fantasma. Si tratta del tasto Ctrl, dei due tasti Shift, dei due tasti Amiga e dei due tasti Alt.

Interfaccia parallela

L'interfaccia parallela bidirezionale presenta un connettore a 25 pin sul retro dell'elaboratore. Viene generalmente usata per il collegamento di una stampante parallela.

Per ogni byte di dati inserito nel registro della porta parallela, l'hardware genera automaticamente il segnale di "dati pronti" (DRDY*) sul pin corrispondente. Il segnale di "dati ricevuti" (ACK*) può dare origine a un interrupt. Per il significato dei pin e la temporizzazione, si consultino le appendici E ed F.

Interfaccia seriale

Un connettore a 25 pin sul retro dell'elaboratore costituisce l'uscita dell'interfaccia seriale. A esso può essere collegata un'ampia gamma di dispositivi, tra cui modem e stampanti seriali.

Per quanto riguarda il significato dei vari pin, si veda l'Appendice E.

INTRODUZIONE ALLA CIRCUITERIA SERIALE

Il chip Paula contiene un Trasmettitore/Ricevitore Asincrono Universale, o UART (Universal Asynchronous Receiver/Transmitter). L'UART è programmabile per qualunque velocità, da 110 bit al secondo a oltre un milione. Può ricevere o trasmettere dati con una lunghezza programmabile di 8 o 9 bit.

L'architettura dell'UART permette un completo controllo software. L'UART è per esempio in grado di scoprire errori di "overrun" (quando i dati esterni giungono tanto velocemente che superano la velocità di rimozione dal registro dei dati ricevuti). Mette anche a disposizione bit di stato e interrupt dedicati alle condizioni di "buffer di ricezione pieno" e "buffer di trasmissione vuoto". Un altro bit di stato indica che tutti i bit di un certo byte di dati hanno subito uno shift verso l'esterno. Tutti questi argomenti sono approfonditi nel prossimo paragrafo.

IMPOSTAZIONE DELLA VELOCITÀ DI TRASMISSIONE E RICEZIONE

La velocità di trasmissione (espressa in baud) è controllata dai bit 14-0 del registro SERPER.

Tutte le temporizzazioni avvengono sulla base dei "clock di colore" che corrispondono a 281,94 ns su una macchina PAL e a 279,36 ns su una macchina NTSC. Se i bit 14-0 del registro SERPER contengono il numero N, allora tra la lettura o la trasmissione di due bit di dati passano N + 1 clock di colore. Il valore da inserire corrisponde quindi alla formula SERPER = (3.546.895/baud) - 1 (PAL) o SERPER = (3.579.545/baud) - 1 (NTSC). Per esempio, il valore di SERPER per una velocità di 9600 baud su una macchina PAL sarà (3.546.895/9.600) - 1 = 368.

Con un cavo di lunghezza ragionevole, la massima velocità affidabile va da 150.000 a 250.000 bit al secondo (il valore varia a seconda dei modelli). A queste velocità non è possibile utilizzare il controllo tramite interrupt a causa del tempo richiesto dal sistema per la loro gestione. Il codice dedicato alla ricezione dev'essere formato da un loop il più breve possibile. Tramite lo scambio

d'informazioni di controllo a bassa velocità e di dati ad alta velocità si può creare un'efficace rete di comunicazione a un costo molto ridotto.

IMPOSTAZIONE DEL MODO DI RICEZIONE

Il numero di bit che devono essere ricevuti prima che il sistema entri in una condizione di "registro di ricezione pieno" varia tra 8 e 9 (il che permette la trasmissione di 8 bit di dati più un bit di parità). In ogni caso, la circuiteria di controllo si aspetta di ricevere un bit di start, otto o nove bit di dati, e almeno un bit di stop.

Il modo di ricezione viene determinato dall'impostazione del bit 15 del registro a sola scrittura SERPER. Il bit va impostato a 1 per ricevere il segnale "registro di ricezione pieno" dopo nove bit di dati, e a 0 per otto bit. Lo stato normale per la maggior parte delle applicazioni è 0.

CONTENUTO DEL REGISTRO DI RICEZIONE DATI

Il registro seriale di ricezione dati contiene 16 bit: 8 o 9 bit di dati più i bit di stato.

I dati vengono ricevuti (un bit per volta) in un registro di shift interno seriale-parallelo. Quando è stato letto il numero atteso di bit, il contenuto di questo registro viene trasferito al registro dati seriale (SERDATR) descritto nella Tavola 8.9, e viene generato un interrupt.

Immediatamente dopo, il registro di shift è pronto ad accettare nuovi dati. Dopo la generazione dell'interrupt, c'è il tempo di ricevere un altro carattere (8-10 bit), per accettare quello precedente e per azzerare il flag di interrupt. Se l'interrupt non viene soddisfatto in tempo, viene impostato il bit di overrun.

Tavola 8.9: I registri SERDATR e ADKCON

SERDATR

Bit	**Nome**	**Funzione**
15	OVRUN	OVERRUN. Questo bit corrisponde al bit 11 (INTF_RBF) di INTREQ. Indica che è stato ricevuto un altro byte di dati prima che la CPU accettasse quello precedente. Per evitare che ciò accada è necessario azzerare in tempo INTF_RBF nel registro INTREQ.
14	RBF	BUFFER DI RICEZIONE PIENO. Questo bit a 1 indica che è presente un byte di dati pronto per essere letto dal 68000. Dopo aver letto il contenuto di questo registro, è necessario azzerare il bit INTF_RBF nel registro INTREQ per evitare una condizione di overrun.
13	TBE	BUFFER DI TRASMISSIONE VUOTO. Questo bit a 1 indica che i dati nel registro di output (SERDAT) sono stati trasferiti al registro di shift seriale e SERDAT è pronto ad accettarne di nuovi. È a 1 anche quando il buffer è vuoto. Viene spesso usato in operazioni full-duplex.

12	TSRE	REGISTRO DI SHIFT VUOTO. Quando questo bit è a 1 il registro di shift in output ha terminato il suo compito: tutti i dati sono stati trasmessi e il registro è inattivo. Se si cessa d'inviare dati a SERDAT, questo bit assume valore 1 solo quando sono state trasmesse sia la word contenuta nel registro di shift sia quella contenuta in SERDAT. Viene spesso usato in operazioni half-duplex.
11	RXD	Collegamento diretto al pin RXD del chip Paula.
10		Non utilizzato.
9	STP	Bit di stop (se sono stati selezionati 9 bit di dati).
8	STP	Bit di stop (se sono stati selezionati 8 bit di dati). OPPURE
	DB8	Nono bit di dati (se sono stati selezionati 9 bit di dati).
7-0	DB7-DB0	8 bit di dati.

ADKCON

15	SET/CLR	Permette d'impostare o azzerare singoli bit.
		Se il bit 15 è a 1, i bit specificati vengono impostati a 1. Se il bit 15 è a 0, i bit specificati vengono impostati a 0.
11	UARTBRK	Forza l'azzeramento del pin di trasmissione.

COME VENGONO TRASMESSI I DATI IN USCITA

I dati in uscita devono essere inseriti nel registro a sola scrittura SERDAT. La velocità a cui vengono inviati è la stessa fissata per la ricezione. Immediatamente dopo la scrittura del registro, inizia la trasmissione dei dati alla velocità prevista.

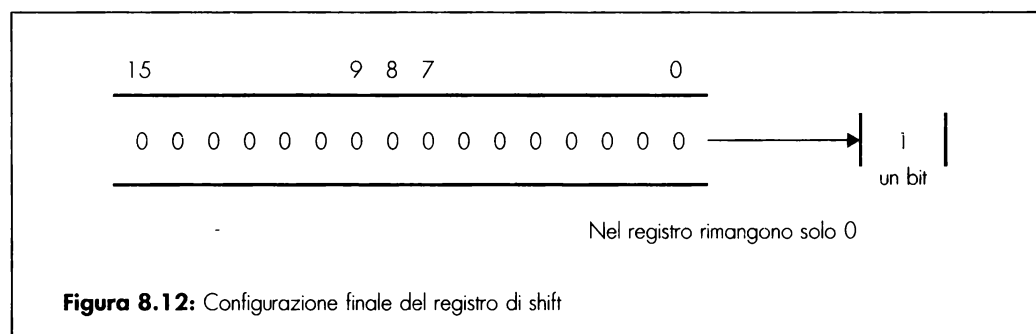
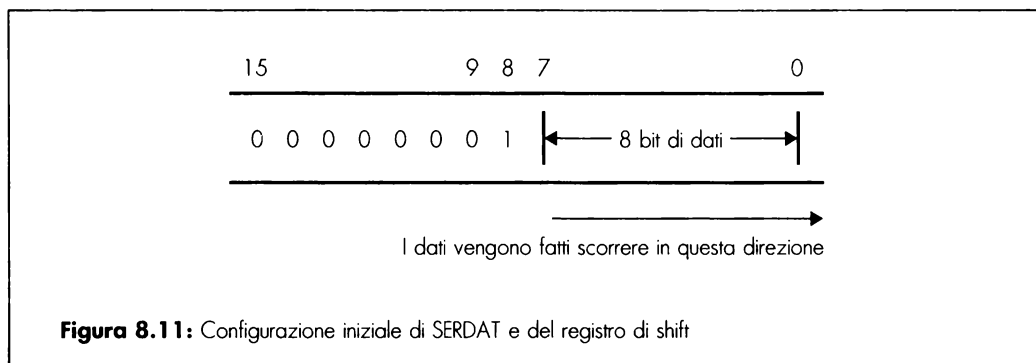
All'inizio dell'operazione, i dati vengono trasferiti in un registro di shift interno. Quando il trasferimento a questo registro è stato completato, il registro SERDAT è pronto ad accettarne altri, e questo viene segnalato dall'interrupt TBE (buffer di trasmissione vuoto).

I dati vengono trasmessi dal registro di shift (un bit per volta) a partire dal bit meno significativo. Lo shift continua finché non vengono trasmessi tutti i bit a 1. in tal modo è possibile selezionare una qualunque combinazione di dati e di bit di stop.

SERDAT è il registro a 16 bit che permette di controllare il formato dei dati trasmessi. Per dare origine a una tipica sequenza di dati, con un bit di start, otto bit di dati e un bit di stop, si studi l'impostazione del registro SERDAT illustrata nelle figure 8.11 e 8.12.

Quando il registro vede che l'ultimo bit trasmesso è un 1 e tutto il resto del registro è a 0, conclude le operazioni e dà origine al segnale TRSE. Quando un nuovo valore diverso da zero viene inserito nel registro, l'operazione di shift ricomincia.

Figura 8.11: Configurazione iniziale di SERDAT e del registro di shift

Figura 8.12: Configurazione finale del registro di shift

COME SPECIFICARE IL CONTENUTO DEL REGISTRO SERDAT

I dati da trasmettere vengono inseriti nel registro SERDAT. In aggiunta ai bit di dati dev'essere aggiunto un bit a 1 come bit di stop. Normalmente vengono inviati uno o due bit di stop.

La trasmissione del bit di start è indipendente dal contenuto del registro: il bit di start viene generato automaticamente prima della trasmissione del primo bit di dati.

Impostare questo registro dà inizio alla trasmissione dei dati. Se tuttavia tutti i bit inseriti sono uguali a zero, non ha luogo nessuna trasmissione.

Connettori di output video

Tutti i modelli di Amiga hanno sul retro dell'unità un connettore a 23 pin, al quale sono collegati i segnali di output video e quelli di input da dispositivi esterni di tipo genlock. Sono disponibili per questo connettore due tipi di output RGB:

- Per monitor RGB (RGB analogico). Si tratta di quattro linee di output: rosso (R), verde (G), blu (B) e sincronismo (S). Attraverso questi segnali è possibile generare 4096 colori diversi sullo schermo utilizzando l'Amiga.

- Monitor RGB digitali. Si tratta di cinque segnali, diversi da quelli visti in precedenza: rosso (R), verde (G), blu (B), bassa intensità (I) e sincronismo (S). Tutti gli ouput sono

livelli logici (0 o 1). Con questo tipo di output, alcuni monitor permettono la generazione di 15 sole combinazioni di colore, poiché i valori 0000 e 0001 corrispondono allo stesso colore (senza colore, infatti, tra bassa e alta intensità non ci sono differenze). Altri monitor associano i 16 valori ad altrettanti colori arbitrari.

Si noti che il segnale di sincronismo dell'Amiga non è associato ad alcun buffer. Per utilizzare dispositivi esterni che presentino un forte carico sul segnale di sincronismo, possono essere necessari buffer esterni.

Gli Amiga 500 e 2000 hanno un jack per uscita video monocromatica, da utilizzare con i monitor più economici. I colori dell'Amiga vengono tradotti in intensità in base alla tavola seguente:

Rosso	Verde	Blu
30%	60%	10%

L'Amiga 1000 possiede un jack di uscita RF. È disponibile un adattatore che permette l'uso dell'Amiga con un normale televisore. Sul jack è disponibile anche il suono stereofonico, ma sul televisore viene generalmente utilizzato un output monofonico.

L'Amiga 1000 possiede inoltre un jack per output videocomposito. Può essere utilizzato per la registrazione diretta tramite videoregistratore, ma la qualità non è elevata.

Nell'uso con un monitor monocromatico le informazioni riguardanti il colore sovente danno origine a effetti indesiderati. Un'attenta scelta del colore o modifiche ai circuiti interni possono migliorare il risultato. Un buon adattatore videocomposito dovrebbe sfruttare la presa RGB a 23 pin.

L'Amiga 2000 possiede uno speciale slot video interno che contiene molti segnali non disponibili altrove: tutti e 23 i segnali della porta RGB, il segnale video digitale non codificato, penna ottica, audio, colore, sincronismo, clock e altri.

A SOMMARIO DEI REGISTRI IN ORDINE ALFABETICO

Questa appendice contiene un completo elenco, in ordine alfabetico, dei registri e dei loro bit.

Gli indirizzi riportati sono quelli usati dai chip custom (Agnus, Denise e Paula) per il reciproco scambio di dati. Anche il Copper, comunque, utilizza questi indirizzi per accedere ai registri dei chip custom. Per utilizzare questi registri con il 68000, se ne può calcolare l'indirizzo utilizzando la seguente formula:

indirizzo del 68000 = indirizzo chip + $DFF000

Per esempio, per accedere al registro ADKCON (indirizzo = $09E), l'indirizzo da usare è $DFF09E. Non esistono altri indirizzi validi. Attenzione: non è consentito accedere a registri non utilizzati.

Tutti i bit dichiarati "non utilizzati" devono essere azzerati in scrittura e ignorati in lettura. I registri sono tutti a sola lettura o a sola scrittura. Leggere un registro a sola scrittura ne distrugge il contenuto. Scrivere in un registro a sola lettura causa effetti non prevedibili.

Tutti i registri di puntamento sono organizzati come 32 bit allineati alle long word. Questi registri possono essere inizializzati con una singola istruzione MOVE.L. Il bit meno significativo di ogni puntatore dev'essere azzerato. I chip custom possono utilizzare soltanto memoria di tipo chip. Tutti i dati relativi ai dischi, agli sprite, ai bitplane, all'audio, alle liste di istruzioni del Copper e qualunque cosa venga sottoposta al Blitter, devono trovarsi nella memoria chip.

Accedendo a un registro strobe, come per esempio COPJMP2 (che risponde sia a una lettura che a una scrittura), assicuratevi di usare l'istruzione MOVE.W e non CLR.W L'istruzione CLR causa una lettura e una scrittura (due accessi) con il 68000, ma solo un accesso con il 68020. Ciò porta a risultati diversi a seconda del microprocessore.

Registro	**Ind.**	**Lett. (R)/ scritt.(W)**	**Agnus/ Denise/ Paula**	**Funzione**
ADKCON	09E	W	P	Controllo dischi e audio, scrittura.
ADKCONR	010	R	P	Controllo dischi e audio, lettura.

BIT	FUNZIONE	
15	SET/CLR	Bit d'impostazione e azzeramento. Determina se i bit a 1 vengono impostati o azzerati. I bit a 0 non vengono modificati.
14-13	PRECOMP 1-0	

CODICE	PRECOMPENSAZIONE
00	nessuna
01	140 ns
10	280 ns
11	560 ns

BIT	NOME	FUNZIONE
12	MFMPREC	1 = MFM, 0 = GCR.
11	UARTBRK	Forza l'azzeramento di TXD.
10	WORDSYNC	Abilita la sincronizzazione della lettura da disco con una word uguale a quella contenuta nel registro DSKSYNC ($07E).
09	MSBSYNC	Abilita la sincronizzazione della lettura da disco con il bit più significativo dei dati (GCR).
08	FAST	Controllo del clock dei dati del disco. 1 = 2μs (MFM), 0 = 4μs (MFM, GCR).
07	USE3PN	Utilizza il canale audio 3 per non modulare nulla.
06	USE2P3	Utilizza il canale audio 2 per modulare il periodo del canale 3.
05	USE1P2	Utilizza il canale audio 1 per modulare il periodo del canale 2.
04	USE0P1	Utilizza il canale audio 0 per modulare il periodo del canale 1.
03	USE3VN	Utilizza il canale audio 3 per non modulare nulla.
02	USE2V3	Utilizza il canale audio 2 per modulare il volume del canale 3.
01	USE1V2	Utilizza il canale audio 1 per modulare il volume del canale 2.
00	USE0V1	Utilizza il canale audio 0 per modulare il volume del canale 1.

NOTA: se nello stesso canale vengono modulati sia il periodo sia il volume, essi saranno alternati. Prima word di dati = V6-V0, seconda word di dati = P15-P0, e così via.

AUDxDAT	0AA	W	P	Dati del canale audio x.

Questo registro costituisce il buffer dei dati DMA del canale audio x (x = 0, 1, 2, 3). Contiene due byte di dati che vengono inviati ai pin di uscita audio in sequenza, con conversione digitale-analogica (LSB = 3 MV). Il controller del DMA trasferisce automaticamente i dati dalla RAM a questo registro. Anche il 68000 può inserire dati direttamente nel registro. Quando termina il trasferimento dei dati tramite DMA (word trasferite = lunghezza), ed è stato utilizzato anche l'ultimo dato trasferito in questo registro, viene generata una richiesta di interrupt.

AUDxLCH	0A0	W	A	Registro di locazione dei dati del canale audio x (4 bit più significativi).
AUDxLCL	0A2	W	A	Registro di locazione dei dati del canale audio x (15 bit meno significativi).

Questa coppia di registri contiene l'indirizzo iniziale dei dati del canale audio x (x = 0, 1, 2, 3). Non si tratta di un registro di puntamento, e pertanto occorre reimpostarlo solo quando si desidera utilizzare dati residenti in un'altra area di memoria.

AUDxLEN	0A4	W	P	Lunghezza dei dati del canale audio x.

Questo registro contiene la lunghezza, espressa in word, dei dati relativi al canale audio x.

AUDxPER	0A6	W	P	Periodo del canale audio x.

Questo registro contiene il periodo (velocità) di trasferimento dati del canale audio x. Il periodo minimo è di 123 clock di colore. Pertanto il valore minimo (decimale) da inserire in questo registro è 123. Ciò corrisponde a una frequenza massima di campionamento pari a 28,86 KHz.

AUDxVOL	0A8	W	P	Volume del canale audio x.

Questo registro specifica il volume del canale audio x. I bit 6-0 possono selezionare uno dei 65 livelli di volume previsti.

Bit	Funzione
15-07	Non utilizzati.
06	Forza il volume al massimo (corrisponde a 64 valori 1 senza nessuno 0).
05-00	Selezionano uno degli altri 64 livelli (000000 = nessun output; 111111 = 63 valori 1 e un valore 0).

BLTAFWM	044	W	A	Maschera della prima word della sorgente A del Blitter.
BLTALWM	046	W	A	Maschera dell'ultima word della sorgente A del Blitter.

I bit contenuti in questi due registri vengono sottoposti a un'operazione di AND logico con la prima e l'ultima word di ogni riga della sorgente A. Nel riempimento di aree e nel tracciamento di linee vengono in genere impostate con tutti 1.

BLTCON0	040	W	A	Registro 0 di controllo del Blitter.
BLTCON1	042	W	A	Registro 1 di controllo del Blitter.

Questi due registri di controllo vengono usati insieme nel controllo delle operazioni del Blitter. Vi sono due modi principali: trasferimento di aree (AREA MODE) e tracciamento di linee (LINE MODE) che vengono selezionati dal bit 0 di BLTCON1, come si vede nelle tabelle seguenti.

AREA MODE (modo normale)

Bit	BLTCON0	BLTCON1
15	ASH3	BSH3
14	ASH2	BSH2
13	ASH1	BSH1
12	ASH0	BSH0
11	USEA	X
10	USEB	X
09	USEC	X
08	USED	X
07	LF7	X
06	LF6	X
05	LF5	X
04	LF4	EFE
03	LF3	IFE
02	LF2	FCI
01	LF1	DESC
00	LF0	LINE (= 0)

ASH3-0	Valore di shift della sorgente A.
BSH3-0	Valore di shift della sorgente B.
USEA	Bit per l'uso della sorgente A.
USEB	Bit per l'uso della sorgente B.
USEC	Bit per l'uso della sorgente C.
USED	Bit per l'uso della destinazione D.
LF7-0	Bit per la selezione dei minterm.
EFE	Abilitazione del fill esclusivo.
IFE	Abilitazione del fill inclusivo.
FCI	Fill carry input (stato iniziale del riempimento per ogni linea).

DESC	Abilitazione del modo discendente.
LINE	Abilitazione del tracciamento di linee (azzerato).

LINE MODE (tracciamento di linee).

BIT	BLTCON0	BLTCON1
15	START3	TEXTURE3
14	START2	TEXTURE2
13	START1	TEXTURE1
12	START0	TEXTURE0
11	1	0
10	0	0
09	1	0
08	1	0
07	LF7	0
06	LF6	SIGN
05	LF5	0 (riservato)
04	LF4	SUD
03	LF3	SUL
02	LF2	AUL
01	LF1	SING
00	LF0	LINE (= 1)

START3-0	Punto iniziale della linea (0-15).
LF7-0	Minterm di selezione della funzione logica.
LINE	Abilitazione del tracciamento di linee (impostato).
SIGN	Flag di segno.
SING	Un solo pixel per linea orizzontale (da utilizzare per riempire l'area in seguito tramite il Blitter).
SUD	In alto o in basso (= AUD*).
SUL	In alto o a sinistra.
AUL	Sempre in alto o a sinistra.

I 3 bit precedenti selezionano l'ottante da utilizzare per tracciare la linea:

OTTANTE	SUD	SUL	AUL
0	1	1	0
1	0	0	1
2	0	1	1
3	1	1	1
4	1	0	1
5	0	1	0
6	0	0	0
7	1	0	0

La sorgente B viene utilizzata per fornire la matrice della linea.

BLTDDAT – – – Registro di destinazione dei dati del Blitter.

Questo registro contiene ogni word restituita dalle operazioni del Blitter finché non viene trasferita alla sua destinazione in RAM. Si tratta di un indirizzo non accessibile al microprocessore. Il trasferimento avviene automaticamente.

BLTSIZE 058 W A Avvio del Blitter e dimensioni del blit (larghezza e altezza).

Questo registro contiene la larghezza e l'altezza dell'area su cui deve operare il Blitter. Scrivervi fa partire il Blitter, e quindi l'operazione di scrittura deve avvenire dopo l'impostazione di tutti gli altri registri.

BIT	15, 14, 13, 12, 11, 10, 09, 08, 07, 06, 05, 04, 03, 02, 01, 00
	h9 h8 h7 h6 h5 h4 h3 h2 h1 h0, w5 w4 w3 w2 w1 w0

h = altezza = numero di linee verticali (10 bit = 1024 linee max).
w = larghezza = numero di word per linea (6 bit = 64 word = 1024 pixel max).

Nel tracciamento di linee, l'altezza corrisponde alla lunghezza della linea (fino a 1024 pixel), e la larghezza deve sempre essere impostata a 02.

BLTxDAT 074 W A Registro dati della sorgente x del Blitter.

Questo registro contiene i dati della sorgente x (x = A, B, C). In genere viene utilizzato dai canali DMA, ma può anche essere impostato dalla CPU.

Nel tracciamento di linee, BLTADAT viene usato come registro indice e dev'essere precaricato con il valore $8000. BLTBDAT viene usato come matrice della linea ($FFFF per una linea continua).

BLTxMOD 064 W A Modulo x del Blitter

Questo registro contiene il valore del modulo per i canali sorgenti (x = A, B, C) e quello destinazione (x = D) del Blitter. Il modulo è un numero che viene aggiunto all'indirizzo al termine di ogni linea, per far sì che punti correttamente alla linea successiva. Ogni canale possiede il proprio modulo.

Nel tracciamento di linee, BLTAMOD e BLTBMOD sono usati come registri per il controllo dell'inclinazione della linea e devono contenere i valori (4Y-4X) e (4Y). Y/X = pendenza della linea. BLTCMOD e BLTDMOD devono contenere la larghezza del bitplane destinazione espressa in byte.

BLTxPTH	050	W	A	Puntatore relativo al canale x del Blitter (4 bit più significativi).
BLTxPTL	052	W	A	Puntatore relativo al canale x del Blitter (15 bit meno significativi).

Questa coppia di registri contiene l'indirizzo dei canali sorgenti (x = A, B, C) e di quello destinazione (x = D) del Blitter. Quando il Blitter ha terminato le sue operazioni, contiene invece l'indirizzo successivo all'ultima word (più il modulo corrispondente).

Nel tracciamento di linee, BLTAPTL viene usato come accumulatore e deve contenere il valore (2Y-X), dove Y/X è la pendenza della linea. BLTCPT e BLTDPT devono contenere l'indirizzo corrispondente al primo pixel della linea.

BPL1MOD	108	W	A	Modulo dei bitplane dispari.
BPL2MOD	10A	W	A	Modulo dei bitplane pari.

Questi registri contengono il modulo dei bitplane pari e dispari. Il modulo è un valore che viene automaticamente aggiunto all'indirizzo al termine di ogni linea, per ricavare il corretto indirizzo della linea successiva.

Dal momento che hanno moduli separati, i bitplane pari e dispari possono avere dimensioni diverse l'uno dall'altro e diverse anche da quelle della finestra video.

BPLCON0	100	W	A/D	Registro di controllo del bitplane 0.
BPLCON1	102	W	D	Registro di controllo del bitplane 1.
BPLCON2	104	W	D	Registro di controllo del bitplane 2.

Questi registri controllano le operazioni dei bitplane e vari aspetti del video.

BIT	BPLCON0	BPLCON1	BPLCON2
15	HIRES	X	X
14	BPU2	X	X
13	BPU1	X	X
12	BPU0	X	X
11	HOMOD	X	X
10	DBLPF	X	X
09	COLOR	X	X
08	GAUD	X	X
07	X	PF2H3	X
06	X	PF2H2	PF2PRI
05	X	PF2H1	PF2P2
04	X	PF2H0	PF2P1
03	LPEN	PF1H3	PF2P0
02	LACE	PF1H2	PF1P2
01	ERSY	PF1H1	PF1P1
00	X	PF1H0	PF1P0

HIRES	Modo ad alta risoluzione (640).
BPU	Numero di bitplane usati (da 0 a 6).
HOMOD	Modo HAM.
DBLPF	Modo dual-playfield.
COLOR	Abilitazione dell'uscita videocomposita.
GAUD	Abilitazione del genlock audio (dirottato sul pin BKGND durante l'intervallo di vertical blanking).
LPEN	Abilitazione della penna ottica (azzerato durante il reset).
LACE	Modo interlace (azzerato durante il reset).
ERSY	Sincronizzazione esterna (HSYNC e VSYNC diventano input) (azzerato durante il reset).
PF2PRI	Il playfield 2 (bitplane pari) ha priorità sul playfield 1 (bitplane dispari).
PF2P	Codice di priorità del playfield 2 rispetto agli sprite.
PF1P	Codice di priorità del playfield 1 rispetto agli sprite.
PF2H	Scroll orizzontale del playfield 2.
PF1H	Scroll orizzontale del playfield 1.

BPLxDAT	110	W	D	Dati del bitplane x (convertitore parallelo-seriale).

Questi registri ricevono i dati dalla RAM tramite i puntatori impiegati dai canali DMA. Possono anche essere impostati dal microprocessore. Essi agiscono come convertitori parallelo-seriali di 6 word, per un massimo di sei bitplane. La conversione inizia nel momento in cui vengono impostati i dati del bitplane 1. La conversione inizia dal bit più significativo, che si trova pertanto sempre a sinistra.

BPLxPTH	0E0	W	A	Puntatore al bitplane x (4 bit più significativi).
BPLxPTL	0E2	W	A	Puntatore al bitplane x (15 bit meno significativi).

Questa coppia di registri contiene l'indirizzo del bitplane x (x = 1, 2, 3, 4, 5, 6). Questo registro dev'essere reinizializzato dal 68000 o dal Copper a ogni intervallo di vertical blanking.

CLXCON	098	W	D	Controllo delle collisioni.

Questo registro controlla quali bitplane devono essere presi in considerazione nella verifica delle collisioni e lo stato richiesto per la collisione. Controlla anche l'inclusione degli sprite dispari tramite un'operazione di OR logico con gli sprite pari.

BIT	NOME	FUNZIONE
15	ENSP7	Abilita lo sprite 7 (OR con sprite 6).
14	ENSP5	Abilita lo sprite 5 (OR con sprite 4).
13	ENSP3	Abilita lo sprite 3 (OR con sprite 2).

		12	ENSP1	Abilita lo sprite 1 (OR con sprite 0).
		11	ENBP6	Abilita il bitplane 6.
		10	ENBP5	Abilita il bitplane 5.
		09	ENBP4	Abilita il bitplane 4.
		08	ENBP3	Abilita il bitplane 3.
		07	ENBP2	Abilita il bitplane 2.
		06	ENBP1	Abilita il bitplane 1.
		05	MVBP6	Valore di verifica per il bitplane 6.
		04	MVBP5	Valore di verifica per il bitplane 5.
		03	MVBP4	Valore di verifica per il bitplane 4.
		02	MVBP3	Valore di verifica per il bitplane 3.
		01	MVBP2	Valore di verifica per il bitplane 2.
		00	MVBP1	Valore di verifica per il bitplane 1.

NOTA: i bitplane disabilitati non possono impedire le collisioni. Pertanto, se tutti i bitplane sono disabilitati, queste si verificano sempre, indipendentemente dal valore dei bit 5-0.

CLXDAT	00E	R	D	Registro dati per le collisioni.

Tramite questo indirizzo vengono letti, e contemporaneamente azzerati, i dati riguardanti le collisioni.

NOTA: il playfield 1 è costituito dai bitplane dispari abilitati, il playfield 2 da quelli pari.

Bit	COLLISIONE REGISTRATA
15	Non utilizzato.
14	Sprite 4 (o 5) e sprite 6 (o 7).
13	Sprite 2 (o 3) e sprite 6 (o 7).
12	Sprite 2 (o 3) e sprite 4 (o 5).
11	Sprite 0 (o 1) e sprite 6 (o 7).
10	Sprite 0 (o 1) e sprite 4 (o 5).
09	Sprite 0 (o 1) e sprite 2 (o 3).
08	Playfield 2 e sprite 6 (o 7).
07	Playfield 2 e sprite 4 (o 5).
06	Playfield 2 e sprite 2 (o 3).
05	Playfield 2 e sprite 0 (o 1).
04	Playfield 1 e sprite 6 (o 7).
03	Playfield 1 e sprite 4 (o 5).
02	Playfield 1 e sprite 2 (o 3).
01	Playfield 1 e sprite 0 (o 1).
00	Playfield 1 e playfield 2.

COLORxx	180	W	D	Registro di colore xx.

Vi sono 32 registri di questo tipo (xx = 00-31), che formano la cosiddetta "palette" di colori. Ciascuno di essi contiene un codice a 12 bit che rappresenta i valori d'intensità per le componenti RGB. Per

ogni pixel sul video, il valore ottenuto dalla combinazione seriale dei registri BPLxDAT seleziona uno di questi registri, il cui contenuto viene inviato ai pin di output RGB.

BIT	15,	14,	13,	12,	11,	10,	09,	08,	07,	06,	05,	04,	03,	02,	01,	00
RGB	X	X	X	X	R3	R2	R1	R0	G3	G2	G1	G0	B3	B2	B1	B0

R = rosso, G = verde, B = blu.

COP1LCH	080	W	A	Primo registro di locazione del Copper (4 bit più significativi).
COP1LCL	082	W	A	Primo registro di locazione del Copper (15 bit meno significativi).
COP2LCH	084	W	A	Secondo registro di locazione del Copper (4 bit più significativi).
COP2LCL	086	W	A	Secondo registro di locazione del Copper (15 bit meno significativi).

Questi registri contengono gli indirizzi iniziali delle liste di istruzioni del Copper.

COPCON	02E	W	A	Registro di controllo del Copper.

Si tratta di un registro da 1 bit che permette al Copper di accedere ai registri del Blitter. Viene azzerato al reset del sistema.

BIT	NOME	FUNZIONE
01	CDANG	Quando è impostato a 1 permette l'accesso al Blitter da parte del Copper.

COPINS	08C	W	A	Registro istruzioni temporaneo.

Questo è un indirizzo fittizio che viene generato dal Copper ogni volta che carica una nuova istruzione nel proprio registro interno. Ciò avviene a ogni ciclo del Copper, fuorché nel secondo ciclo (IR2) dell'istruzione MOVE. Le tre istruzioni disponibili sono illustrate qui di seguito.

MOVE	Movimento immediato di dati a un registro destinazione.
WAIT	Attende che il pennello elettronico raggiunga una specifica posizione sul video (finché la posizione non viene raggiunta, il Copper non utilizza il bus dati).
SKIP	Salta l'istruzione successiva se il pennello elettronico ha già raggiunto una specifica posizione sul video.

	MOVE		WAIT		SKIP	
BIT	IR1	IR2	IR1	IR2	IR1	IR2
15	X	RD15	VP7	BFD*	VP7	BFD*
14	X	RD14	VP6	VE6	VP6	VE6
13	X	RD13	VP5	VE5	VP5	VE5
12	X	RD12	VP4	VE4	VP4	VE4
11	X	RD11	VP3	VE3	VP3	VE3
10	X	RD10	VP2	VE2	VP6	VE2
09	X	RD09	VP1	VE1	VP6	VE1
08	DA8	RD08	VP0	VE0	VP6	VE0
07	DA7	RD07	HP8	HE8	HP8	HE8
06	DA6	RD06	HP7	HE7	HP7	HE7
05	DA5	RD05	HP6	HE6	HP6	HE6
04	DA4	RD04	HP5	HE5	HP5	HE5
03	DA3	RD03	HP4	HE4	HP4	HE4
02	DA2	RD02	HP3	HE3	HP3	HE3
01	DA1	RD01	HP2	HE2	HP2	HE2
00	0	RD00	1	0	1	1

IR1 Prima word dell'istruzione.
IR2 Seconda word dell'istruzione.
DA Registro destinazione dell'istruzione MOVE. Viene letto durante IR1, e utilizzato durante IR2 sul bus RGA.
RD Dato immediato usato dall'istruzione MOVE. Viene trasferito direttamente dalla RAM al registro specificato in IR1.
VP Bit di confronto della posizione verticale.
HP Bit di confronto della posizione orizzontale.
VE Bit di abilitazione del confronto verticale.
HE Bit di abilitazione del confronto orizzontale.

* BFD = Bit di attesa del Blitter. Quando questo bit è impostato il flag di blit concluso non ha alcun effetto sul Copper. Quando invece è a zero, il flag di blit concluso dev'essere a 1 (oltre a tutti i confronti di posizione), perché il Copper possa uscire dallo stato di attesa in un'istruzione WAIT o SKIP. Si noti che per questo motivo il bit di confronto V7 non può essere mascherato.

COPJMP1 088 W A Riporta il Copper alla locazione 1.
COPJMP2 08A W A Riporta il Copper alla locazione 2.

Questi sono indirizzi di tipo strobe. Un qualunque accesso a questi registri causa il salto indiretto del Copper all'istruzione contenuta in uno dei due registri di locazione. Lo stesso Copper può accedere a questi due registri, causando il proprio salto indiretto.

DDFSTOP 094 W A Coordinata orizzontale finale del trasferimento dati.

DDFSTRT	092	W	A	Coordinata orizzontale iniziale del trasferimento dati.

Questi registri controllano la temporizzazione orizzontale di inizio e fine del trasferimento dati per ogni linea del video. In verticale è invece uguale a quella della finestra video. Il modulo dei bitplane dipende dalle dimensioni degli stessi bitplane e dal contenuto di questi registri.

BIT	15,	14,	13,	12,	11,	10,	09,	08,	07,	06,	05,	04,	03,	02,	01,	00
FUNZ.	X	X	X	X	X	X	X	X	H8	H7	H6	H5	H4	H3	X	X

I bit contrassegnati con la X devono essere impostati a zero, per garantire la compatibilità con versioni future.

DDFSTRT (limite sinistro del trasferimento dati).

DIM. SCHERMO	H8	H7	H6	H5	H4
Massimo	0	0	1	0	1
Largo	0	0	1	1	0
Normale	0	0	1	1	1
Stretto	0	1	0	0	0

DDFSTOP (limite destro del trasferimento dati).

DIM. SCHERMO	H8	H7	H6	H5	H4
Stretto	1	1	0	0	1
Normale	1	1	0	1	0
Massimo	1	1	0	1	1

DIWSTOP	090	W	A	Limite inferiore destro della finestra video.
DIWSTRT	08E	W	A	Limite superiore sinistro della finestra video.

Questi registri controllano le dimensioni e la posizione della finestra video tramite gli angoli superiore sinistro e inferiore destro.

BIT	15,	14,	13,	12,	11,	10,	09,	08,	07,	06,	05,	04,	03,	02,	01,	00
FUNZ.	V7	V6	V5	V4	V3	V2	V1	V0	H7	H6	H5	H4	H3	H2	H1	H0

DIWSTRT è ristretto verticalmente alla metà superiore dello schermo (V8 = 0) e orizzontalmente ai 3/4 di sinistra dello schermo (H8 = 0).

DIWSTOP è ristretto verticalmente alla metà inferiore dello schermo (V8 = /V7) e orizzontalmente all'ultimo quarto dello schermo (H8 = 1).

DMACON	096	W	A/P/D	Controllo del DMA – scrittura (impostazione o azzeramento).
DMACONR	002	R	A/P	Controllo del DMA e stato del Blitter – lettura.

Questo registro contiene i bit di controllo di tutti i canali DMA e i bit di stato del Blitter.

BIT	NOME	FUNZIONE
15	SET/CLR	Bit d'impostazione/azzeramento. Determina se i bit a 1 vengono impostati o azzerati. I bit a 0 non vengono modificati.
14	BBUSY	Bit di Blitter occupato (a sola lettura).
13	BZERO	Bit di zero del Blitter (a sola lettura).
12	X	
11	X	
10	BLTPRI	Bit di priorità del Blitter. Disabilita il pin /BLS, dando al Blitter completa priorità sul 68000.
09	DMAEN	Abilita tutti i seguenti canali DMA.
08	BPLEN	Abilita il DMA dei bitplane.
07	COPEN	Abilita il DMA del Copper.
06	BLTEN	Abilita il DMA del Blitter.
05	SPREN	Abilita il DMA degli sprite.
04	DSKEN	Abilita il DMA dei dischi.
03	AUD3EN	Abilita il DMA del canale audio 3.
02	AUD2EN	Abilita il DMA del canale audio 2.
01	AUD1EN	Abilita il DMA del canale audio 1.
00	AUD0EN	Abilita il DMA del canale audio 0.

DSKBYTR	01A	R	P	Registro di lettura di dati da disco.

Questo registro contiene il buffer dati del microprocessore per la lettura da disco. Durante la lettura, i dati provenienti dal disco passano attraverso questo registro un byte alla volta, causando l'impostazione del bit 15 (DSKBYT).

BIT	NOME	FUNZIONE
15	DSKBYT	Il registro contiene un nuovo byte di dati (azzerato dalla lettura).
14	DMAON	Specchio del bit 15 di DSKLEN (DMAEN) dopo un'operazione di AND con il bit 9 di DMACON (DMAEN).
13	DISKWRITE	Specchio del bit 14 (WRITE) di DSKLEN.
12	WORDEQUAL	Questo bit è a 1 solo quando il registro DSKSYNC corrisponde ai dati letti dal disco.
11-08	X	Non utilizzati.
07-00		Byte di dati proveniente dal disco.

DSKDAT	026	W	P	Dati in scrittura del DMA dei dischi.
DSKDATR	008	ER	P	Dati in lettura del DMA dei dischi.

Questo registro costituisce il buffer dati del DMA dei dischi. Contiene due byte di dati inviati o ricevuti dal disco. La direzione dei dati (lettura o scrittura) è controllata dal bit 14 del registro DSKLEN. Il controller del DMA trasferisce automaticamente i dati da questo registro alla RAM e viceversa, e quando il trasferimento dei dati è finito, genera una richiesta di interrupt.

DSKLEN	024	W	P	Lunghezza dei dati DMA per il disco.

Questo registro contiene la lunghezza, espressa in word, dei dati da trasferire verso il disco o dal disco tramite il relativo canale DMA. Contiene inoltre due bit di controllo: un bit di abilitazione del DMA e un bit per la direzione del DMA (lettura o scrittura).

BIT	NOME	FUNZIONE
15	DMAEN	Abilitazione del DMA del disco.
14	WRITE	A 1 seleziona la scrittura da RAM a disco.
13-00	LENGTH	Lunghezza in word dei dati.

DSKPTH	020	W	A	Puntatore ai dati del disco (4 bit più significativi).
DSKPTL	022	W	A	Puntatore ai dati del disco (15 bit meno significativi).

Questa coppia di registri contiene l'indirizzo dell'area dati che dev'essere utilizzata nell'operazione di trasferimento dati dal disco o verso il disco. Devono essere impostati dal 68000 o dal Copper prima dell'abilitazione del DMA.

DSKSYNC	07E	W	P	Registro di sincronizzazione dei dischi.

Contiene la word di sincronizzazione per la lettura da disco. Vedere anche il bit 10 di ADKCON.

INTENA	09A	W	P	Bit di abilitazione degli interrupt (impostazione o azzeramento).
INTENAR	01C	R	P	Bit di abilitazione degli interrupt (lettura).

Questo registro contiene i bit di abilitazione degli interrupt. Di seguito viene riportato il significato dei singoli bit, valido anche per il registro INTREQ.

BIT	NOME	LIVELLO	FUNZIONE
15	SET/CLR		Bit d'impostazione/azzeramento. Determina se i bit a 1 vengono impostati o azzerati. I bit a 0 non vengono modificati.
14	INTEN		Master interrupt (solo abilitazione, non richiesta).
13	EXTER	6	Interrupt esterno.
12	DSKSYN	5	Il registro DSKSYNC corrisponde ai dati letti dal disco.
11	RBF	5	Buffer di ricezione della porta seriale pieno.
10	AUD3	4	Termine dei dati del canale audio 3.
09	AUD2	4	Termine dei dati del canale audio 2.
08	AUD1	4	Termine dei dati del canale audio 1.
07	AUD0	4	Termine dei dati del canale audio 0.
06	BLIT	3	Blit concluso.
05	VERTB	3	Inizio dell'intervallo di vertical blanking.
04	COPER	3	Copper.
03	PORTS	2	Porte di I/O e timer.
02	SOFT	1	Riservato a interrupt software.
01	DSKBLK	1	Termine del blocco di dati del disco.
00	TBE	1	Buffer di trasmissione della porta seriale vuoto.

INTREQ	09C	W	P	Bit di richiesta di interrupt (impostazione o azzeramento).
INTREQR	01E	R	P	Bit di richiesta di interrupt (lettura).

Questo registro contiene i bit che indicano le richieste di interrupt, modificabili dalla CPU. Se i corrispondenti bit di abilitazione sono impostati, causano l'interrupt relativo. Questi bit non sono automaticamente azzerati al termine dell'interrupt, ma vanno azzerati esplicitamente accedendo a questo indirizzo. L'assegnazione dei bit è la stessa del registro di abilitazione visto in precedenza.

JOY0DAT	00A	R	D	Dati del mouse/joystick 0.
JOY1DAT	00C	R	D	Dati del mouse/joystick 1.

Ognuno di questi registri contiene una coppia di contatori a 8 bit per il mouse. 0 = coppia di sinistra, 1 = coppia di destra, 4 contatori in totale. L'uso dei bit di entrambi gli indirizzi è illustrato qui di seguito. Ogni contatore è controllato dai segnali provenienti da due pin della porta. Per determinare lo stato di questi due pin si possono esaminare i bit 0 e 1 di ogni contatore. Ciò permette di utilizzarli anche con il joystick.

Uso dei contatori con il mouse:
(pin 1,3 = Yclock, pin 2,4 = Xclock)

BIT	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0DAT	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	X7	X6	X5	X4	X3	X2	X1	X0
1DAT	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	X7	X6	X5	X4	X3	X2	X1	X0

La seguente tabella descrive il significato dei pin di connessione con mouse e joystick. La condizione dei pin viene trasferita nei registri all'interno del chip Denise durante il relativo periodo di clock. Si noti che le funzioni del joystick sono tutte "attive basse" ai pin di connessione.

			Trasferiti in Denise		
Pin	Joystick	Mouse	Pin	Nome	Clock
L1	FORW*	Y	38	M0V	CCK
L2	LEFT*	YQ	38	M0V	CCK*
L3	BACK*	X	9	M0H	CCK
L4	RIGH*	XQ	9	M0H	CCK*
R1	FORW*	Y	39	M1V	CCK
R2	LEFT*	YQ	39	M1V	CCK*
R3	BACK*	X	8	M1H	CCK
R4	RIGH*	XQ	8	M1H	CCK*

Dopo il campionamento, i segnali provenienti da questi pin vengono utilizzati nella quadratura per la temporizzazione dei contatori del mouse. Le funzioni LEFT e RIGHT del joystick (sinistra e destra), sono direttamente disponibili nei bit X1 e Y1 di ogni registro. Per ricavare lo stato delle funzioni FORWARD e BACK (avanti e indietro), invece, è necessario combinare tramite un'operazione di OR esclusivo i due bit meno significativi di ogni contatore:

Avanti	Y1 XOR Y0 (bit 09 XOR bit 08)
Sinistra	Y1
Indietro	X1 XOR X0 (bit 01 XOR bit 00)
Destra	X1

JOYTEST	036	W	D	Imposta contemporaneamente i quattro contatori mouse/joystick.

Dati inseriti nei registri:

BIT	15 14 13 12 11 10 09 08	07 06 05 04 03 02 01 00
0DAT	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	X7 X6 X5 X4 X3 X2 X1 X0
1DAT	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	X7 X6 X5 X4 X3 X2 X1 X0

POT0DAT	012	R	P	Coppia di contatori proporzionali di sinistra.
POT1DAT	014	R	P	Coppia di contatori proporzionali di destra.

Questi indirizzi contengono ognuno una coppia di contatori a 8 bit collegati a un potenziometro. Il significato dei singoli bit è riassunto nella tabella che segue.

BIT	15 14 13 12 11 10 09 08	07 06 05 04 03 02 01 00
RIGHT	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	X7 X6 X5 X4 X3 X2 X1 X0
LEFT	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	X7 X6 X5 X4 X3 X2 X1 X0

CONNETTORE			PAULA		
Loc.	Dir.	Sym	Pin	Pin	Nome
RIGHT	Y	RY	9	36	(POT1Y)
RIGHT	X	RX	5	35	(POT1X)
LEFT	Y	LY	9	33	(POT0Y)
LEFT	X	LX	5	32	(POT0X)

POTGO	034	W	P	Dati della porta pot (scrittura e avvio).
POTGOR	016	R	P	Dati della porta pot (lettura). In precedenza era chiamato POTINP.

Questo registro controlla una porta bidirezionale di I/O a 4 bit che utilizza i medesimi quattro pin dei quattro contatori visti in precedenza.

BIT	NOME	FUNZIONE
15	OUTRY	Abilita l'output sul pin 36 di Paula.
14	DATRY	Dati di I/O sul pin 36 di Paula.
13	OUTRX	Abilita l'output sul pin 35 di Paula.
12	DATRX	Dati di I/O sul pin 35 di Paula.
11	OUTLY	Abilita l'output sul pin 33 di Paula.
10	DATLY	Dati di I/O sul pin 33 di Paula.
09	OUTLX	Abilita l'output sul pin 32 di Paula.
08	DATLX	Dati di I/O sul pin 32 di Paula.
07-01	0	Codice di identificazione del chip (attualmente 0).
00	START	Avvia i potenziometri (scarica i condensatori e azzera i contatori).

REFPTR 028 W A Puntatore per il refresh.

Questo registro viene usato come generatore dinamico di indirizzi per il refresh della memoria. È accessibile soltanto a scopo di test e non dovrebbe essere mai modificato dal 68000.

SERDAT 030 W P Buffer di trasmissione dati della porta seriale.

Questo registro costituisce il buffer di trasmissione dati della porta seriale. I dati vengono trasferiti da qui al registro di shift seriale ogni volta che quest'ultimo risulta vuoto. Ciò dà origine alla richiesta di un interrupt di tipo TBE (buffer di trasmissione vuoto). Nei dati dev'essere incluso un bit di stop. La lunghezza dei dati dipende anche dalla posizione del bit di stop.

BIT	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
USO	0	0	0	0	0	0	S	D8	D7	D6	D5	D4	D3	D2	D1	D0

Nota: S = Bit di stop = 1, D = bit di dati

SERDATR 018 R P Buffer di ricezione dati della porta seriale.

I dati contenuti in questo registro arrivano dal registro di shift seriale ogni volta che quest'ultimo è pieno. Oltre ai dati, a questo indirizzo si possono leggere diversi bit di richiesta di interrupt e altri bit di stato.

BIT	NOME	FUNZIONE
15	OVRUN	Overrun in ricezione alla porta seriale. Si azzera ponendo a zero il bit 11 di INTREQ.
14	RBF	Buffer di ricezione pieno.
13	TBE	Buffer di trasmissione vuoto.
12	TSRE	Registro di shift seriale di trasmissione vuoto.
11	RXD	Collegamento diretto con il pin RXD dell'UART per una lettura tramite la CPU.
10	0	Non utilizzato.
09	STP	Bit di stop.
08	STP/DB8	Bit di stop o nono bit di dati.
07-00	DB7-DB0	Bit di dati.

SERPER 032 W P Velocità e controllo della porta seriale.

Questo registro contiene il bit di definizione della lunghezza dei dati e un numero a 15 bit che descrive la velocità di trasmissione e ricezione assegnata alla porta seriale. Se chiamiamo questo numero "N", la velocità risulta di 1 bit ogni (N + 1) * 0,2819 microsecondi (PAL).

BIT	NOME	FUNZIONE
15	LONG	Imposta la lunghezza dei dati a 9 bit.
14-00	RATE	Imposta la velocità a 1/((N + 1) * 0,2819 microsecondi).

SPRxCTL	142	W	A/D	Posizione di stop verticale e bit di controllo dello sprite x.
SPRxPOS	140	W	A/D	Posizione iniziale dello sprite x (verticale e orizzontale).

Questi due registri contengono tutte le informazioni relative alla posizione, alle dimensioni e al controllo dello sprite x. Sono in genere impostate dal DMA relativo allo sprite, durante l'intervallo di horizontal blanking; tuttavia possono essere modificate in qualunque momento dalla CPU.

Registro SPRxPOS:

BIT	NOME	FUNZIONE
15-08	SV7-SV0	Coordinata iniziale verticale. Il bit più significativo (SV8) è contenuto nel registro SPRxCTL.
07-00	SH8-SH1	Coordinata iniziale orizzontale. Il bit meno significativo (SH0) è contenuto nel registro SPRxCTL.

Registro SPRxCTL: (accedere a questo indirizzo disabilita il circuito di confronto orizzontale dello sprite, impedendone così la visualizzazione)

BIT	NOME	FUNZIONE
15-08	EV7-EV0	Gli otto bit meno significativi della coordinata finale verticale dello sprite.
07	ATT	Bit di collegamento dello sprite (per sprite dispari).
06-03	X	Non utilizzati.
02	SV8	Bit più significativo della coordinata iniziale verticale.
01	EV8	Bit più significativo della coordinata finale verticale.
00	SH0	Bit meno significativo della coordinata orizzontale.

SPRxDATA	144	W	D	Registro dati A dello sprite x.
SPRxDATB	146	W	D	Registro dati B dello sprite x.

Questi registri contengono i dati relativi all'immagine dello sprite. Vengono generalmente impostati dal DMA del canale corrispondente, ma possono essere modificati in qualunque momento dal 68000. Quando il confronto con la posizione orizzontale dà un risultato positivo, il contenuto di questi registri viene inviato ai registri di shift, e di qui allo schermo, a partire dal bit più significativo.

Nota: l'accesso al registro dati A abilita lo sprite. Accedere al registro SPRxCTL invece lo disabilita. Quando sono abilitati, i dati contenuti nei registri A e B vengono inviati al video ogni volta che il pennello elettronico raggiunge la posizione orizzontale specificata nel registro SPRxPOS.

SPRxPOS Vedere SPRxCTL.

SPRxPTH	120	W	A	Indirizzo dei dati relativi allo sprite x (4 bit più significativi).
SPRxPTL	122	W	A	Indirizzo dei dati relativi allo sprite x (15 bit meno significativi).

Questa coppia di registri contiene l'indirizzo dei dati relativi allo sprite x (x = 0, 1, 2, 3, 4, 5, 6, 7). Questi registri devono essere inizializzati dal microprocessore o dal Copper durante ogni intervallo di vertical blanking.

STREQU	038	S	D	Registro strobe per la sincronizzazione orizzontale con VB ed EQU.
STRHOR	03C	S	D/P	Registro strobe per la sincronizzazione orizzontale.
STRLONG	03E	S	D	Registro strobe per l'identificazione di una linea orizzontale lunga.

Uno degli indirizzi dei primi tre registri viene collocato sul bus degli indirizzi di destinazione durante il primo slot di refresh. Il quarto registro viene utilizzato ogni due linee orizzontali, durante il secondo slot di refresh, per identificare le linee lunghe (228). Vi sono quattro slot di refresh e quelli non usati per i registri strobe lasciano un indirizzo nullo ($FF) sul bus degli indirizzi di destinazione.

STRVBL	03A	S	D	Registro strobe per la sincronizzazione orizzontale con VB.
VHPOSR	006	R	A	Lettura delle coordinate orizzontali e verticali del pennello elettronico o della penna ottica.
VHPOSW	02C	W	A	Scrittura delle coordinate orizzontali e verticali del pennello elettronico o della penna ottica.

BIT	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FUNZ.	V7	V6	V5	V4	V3	V2	V1	V0	H8	H7	H6	H5	H4	H3	H2	H1

Risoluzione: 1/160 della larghezza dello schermo.

VPOSR	004	R	A	Lettura del bit più significativo della coordinata verticale, e tipo di quadro.
VPOSW	02A	W	A	Scrittura del bit più significativo della coordinata verticale, e tipo di quadro.

BIT	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FUNZ.	LOF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	V8

LOF = quadro lungo.

B SOMMARIO DEI REGISTRI IN ORDINE PROGRESSIVO

In questa appendice vengono utilizzate le seguenti abbreviazioni:

&	Registro usato soltanto dai canali DMA.
%	Registro usato generalmente dai canali DMA, ma anche dalla CPU.
+	Coppia di registri contenenti un indirizzo. Si deve trattare di un indirizzo pari, contenuto nella memoria chip.
*	Registro non accessibile al Copper.
~	Registro non accessibile al Copper se il bit CDANG è a zero.
A, D, P	A = Agnus, D = Denise, P = Paula.
W, R	W = a sola scrittura, R = a sola lettura.
ER	Lettura anticipata. Si tratta di un trasferimento di dati tramite DMA verso la memoria, proveniente dal disco o dal Blitter. La temporizzazione della RAM richiede che i dati si trovino sul bus prima dei cicli di lettura del microprocessore. Questi trasferimenti vengono perciò iniziati secondo la temporizzazione del chip Agnus, e non tramite un indirizzo di lettura sul bus degli indirizzi di destinazione.
S	Strobe (indirizzo di scrittura senza bit assegnati). Se si accede al registro, viene causato l'effetto corrispondente.

PTL, PTH Puntatori dinamici a RAM di tipo chip per il trasferimento di dati tramite DMA. Questi registri devono essere sempre impostati *prima* dell'uso (per bitplane e sprite durante l'intervallo di vertical blanking, per il Blitter prima d'iniziare il blit).

LCL, LCH Indirizzi iniziali di blocchi di RAM di tipo chip da utilizzare per trasferirvi dati tramite DMA. Vengono impiegati per reimpostare automaticamente puntatori dinamici come il program counter del Copper (durante l'intervallo di vertical blanking) e i contatori dei dati dei canali audio (quando il conteggio precedente raggiunge lo zero).

MOD Modulo a 15 bit. Un numero che viene automaticamente aggiunto a un indirizzo al termine di ogni linea in modo da ottenere il corretto indirizzo iniziale per la linea successiva. Ciò permette al Blitter o alla finestra video di operare su blocchi rettangolari di dati di dimensioni inferiori a quelle dell'intero bitplane. Utilizzano 15 bit più l'estensione del segno.

NOME	INDIRIZZO	R/W	CHIP	FUNZIONE
BLTDDAT	&*000	ER	A	Lettura anticipata per la destinazione del Blitter (indirizzo fittizio).
DMACONR	*002	R	A D P	Lettura del controllo del DMA e dello stato del Blitter.
VPOSR	*004	R	A	Bit più significativo della posizione verticale e indicatore di quadro lungo o breve.
VHPOSR	*006	R	A	Posizione orizzontale e verticale.
DSKDATR	&*008	ER	P	Prima lettura dei dati da disco (indirizzo fittizio).
JOY0DAT	*00A	R	D	Dati del mouse/joystick 0.
JOY1DAT	*00C	R	D	Dati del mouse/joystick 1.
CLXDAT	*00E	R	D	Registro dati di collisione (lettura e cancellazione).
ADKCONR	*010	R	P	Lettura del controllo dischi e audio.
POT0DAT	*012	R	P	Coppia di contatori pot 0.
POT1DAT	*014	R	P	Coppia di contatori pot 1.
POTGOR	*016	R	P	Lettura dei dati della porta pot (detto anche POTINP).
SERDATR	*018	R	P	Lettura dati e stato della porta seriale.
DSKBYTR	*01A	R	P	Lettura dati e stato del disco.
INTENAR	*01C	R	P	Lettura dei bit di abilitazione degli interrupt.
INTREQR	*01E	R	P	Lettura dei bit per la richiesta di interrupt.
DSKPTH	+*020	W	A	Puntatore per il DMA del disco (4 bit più significativi).
DSKPTL	+*022	W	A	Puntatore per il DMA del disco (15 bit meno significativi).
DSKLEN	*024	W	P	Lunghezza dei dati del disco.
DSKDAT	&*026	W	P	Scrittura dei dati per il disco.
REFPTR	&*028	W	A	Puntatore per il refresh della memoria.
VPOSW	*02A	W	A	Scrittura del bit più significativo della coordinata verticale e del tipo di quadro video.
VHPOSW	*02C	W	A	Scrittura delle coordinate orizzontale e verticale.
COPCON	*02E	W	A	Registro di controllo del Copper (CDANG).
SERDAT	*030	W	P	Scrittura dati e bit di stop della porta seriale.
SERPER	*032	W	P	Velocità e controllo della porta seriale.
POTGO	*034	W	P	Scrittura e avvio della porta pot.

JOYTEST	*036	W		D		Test dei quattro contatori mouse/joystick.
STREQU	&*038	S		D		Strobe per la sincronizzazione con VB ed EQU.
STRVBL	&*03A	S		D		Strobe per la sincronizzazione con VB (vertical blanking).
STRHOR	&*03C	S		D	P	Strobe per la sincronizzazione orizzontale.
STRLONG	&*03E	S		D		Strobe per l'identificazione di una linea orizzontale lunga.
BLTCON0	~040	W	A			Registro 0 di controllo del Blitter.
BLTCON1	~042	W	A			Registro 1 di controllo del Blitter.
BLTAFWM	~044	W	A			Maschera della prima word della sorgente A.
BLTALWM	~046	W	A			Maschera dell'ultima word della sorgente A.
BLTCPTH	+~048	W	A			Puntatore alla sorgente C del Blitter (4 bit più significativi).
BLTCPTL	+~04A	W	A			Puntatore alla sorgente C del Blitter (15 bit meno significativi).
BLTBPTH	+~04C	W	A			Puntatore alla sorgente B del Blitter (4 bit più significativi).
BLTBPTL	+~04E	W	A			Puntatore alla sorgente B del Blitter (15 bit meno significativi).
BLTAPTH	+~050	W	A			Puntatore alla sorgente A del Blitter (4 bit più significativi).
BLTAPTL	+~052	W	A			Puntatore alla sorgente A del Blitter (15 bit meno significativi).
BLTDPTH	+~054	W	A			Puntatore alla destinazione D del Blitter (4 bit più significativi).
BLTDPTL	+~056	W	A			Puntatore alla destinazione D del Blitter (15 bit meno significativi).
BLTSIZE	~058	W	A			Avvio e dimensioni del blit.
	~05A					
	~05C					
	~05E					
BLTCMOD	~060	W	A			Modulo della sorgente C del Blitter.
BLTBMOD	~062	W	A			Modulo della sorgente B del Blitter.
BLTAMOD	~064	W	A			Modulo della sorgente A del Blitter.
BLTDMOD	~066	W	A			Modulo della destinazione D del Blitter.
	~068					
	~06A					
	~06C					
	~06E					
BLTCDAT	%~070	W	A			Registro dati della sorgente C del Blitter.
BLTBDAT	%~072	W	A			Registro dati della sorgente B del Blitter.
BLTADAT	%~074	W	A			Registro dati della sorgente A del Blitter.
	~076					
	~078					
	~07A					
	~07C					
DSKSYNC	~07E	W			P	Word di sincronizzazione per la lettura dal disco.
COP1LCH	+080	W	A			Primo registro di locazione del Copper (4 bit più significativi).
COP1LCL	+082	W	A			Primo registro di locazione del Copper (15 bit meno significativi).

COP2LCH	+084	W	A	Secondo registro di locazione del Copper (4 bit più significativi).
COP2LCL	+086	W	A	Secondo registro di locazione del Copper (15 bit meno significativi).
COPJMP1	088	S	A	Riporta il PC del Copper all'indirizzo contenuto in COP1LC.
COPJMP2	08A	S	A	Riporta il PC del Copper all'indirizzo contenuto in COP2LC.
COPINS	08C	W	A	Identificazione dell'istruzione letta dal Copper.
DIWSTRT	08E	W	A	Inizio della finestra video.
DIWSTOP	090	W	A	Termine della finestra video.
DDFSTRT	092	W	A	Inizio del trasferimento dati dai bitplane.
DDFSTOP	094	W	A	Termine del trasferimento dati dai bitplane.
DMACON	096	W	A D P	Controllo del DMA.
CLXCON	098	W	D	Controllo delle collisioni.
INTENA	09A	W	P	Bit di abilitazione degli interrupt.
INTREQ	09C	W	P	Bit di richiesta di interrupt.
ADKCON	09E	W	P	Controllo disco, audio, UART.
AUD0LCH	+0A0	W	A	Registro locazione dati del canale audio 0 (4 bit più significativi).
AUD0LCL	+0A2	W	A	Registro locazione dati del canale audio 0 (15 bit meno significativi).
AUD0LEN	0A4	W	P	Lunghezza dati per il canale audio 0.
AUD0PER	0A6	W	P	Periodo del canale audio 0.
AUD0VOL	0A8	W	P	Volume del canale audio 0.
AUD0DAT	&0AA	W	P	Dati del canale audio 0.
	0AC			
	0AE			
AUD1LCH	+0B0	W	A	Registro locazione dati del canale audio 1 (4 bit più significativi).
AUD1LCL	+0B2	W	A	Registro locazione dati del canale audio 1 (15 bit meno significativi).
AUD1LEN	0B4	W	P	Lunghezza dati per il canale audio 1.
AUD1PER	0B6	W	P	Periodo del canale audio 1.
AUD1VOL	0B8	W	P	Volume del canale audio 1.
AUD1DAT	&0BA	W	P	Dati del canale audio 1.
	0BC			
	0BE			
AUD2LCH	+0C0	W	A	Registro locazione dati del canale audio 2 (4 bit più significativi).
AUD2LCL	+0C2	W	A	Registro locazione dati del canale audio 2 (15 bit meno significativi).
AUD2LEN	0C4	W	P	Lunghezza dati per il canale audio 2.
AUD2PER	0C6	W	P	Periodo del canale audio 2.
AUD2VOL	0C8	W	P	Volume del canale audio 2.
AUD2DAT	&0CA	W	P	Dati del canale audio 2.
	0CC			
	0CE			
AUD3LCH	+0D0	W	A	Registro locazione dati del canale audio 3 (4 bit più significativi).
AUD3LCL	+0D2	W	A	Registro locazione dati del canale audio 3 (15 bit meno

						significativi).
AUD0LEN	0D4	W			P	Lunghezza dati per il canale audio 3.
AUD3PER	0D6	W			P	Periodo del canale audio 3.
AUD3VOL	0D8	W			P	Volume del canale audio 3.
AUD3DAT	&0DA	W			P	Dati del canale audio 3.
	0DC					
	0DE					
BPL1PTH	+0E0	W	A			Puntatore al bitplane 1 (4 bit più significativi).
BPL1PTL	+0E2	W	A			Puntatore al bitplane 1 (15 bit meno significativi).
BPL2PTH	+0E4	W	A			Puntatore al bitplane 2 (4 bit più significativi).
BPL2PTL	+0E6	W	A			Puntatore al bitplane 2 (15 bit meno significativi).
BPL3PTH	+0E8	W	A			Puntatore al bitplane 3 (4 bit più significativi).
BPL3PTL	+0EA	W	A			Puntatore al bitplane 3 (15 bit meno significativi).
BPL4PTH	+0EC	W	A			Puntatore al bitplane 4 (4 bit più significativi).
BPL4PTL	+0EE	W	A			Puntatore al bitplane 4 (15 bit meno significativi).
BPL5PTH	+0F0	W	A			Puntatore al bitplane 5 (4 bit più significativi).
BPL5PTL	+0F2	W	A			Puntatore al bitplane 5 (15 bit meno significativi).
BPL6PTH	+0F4	W	A			Puntatore al bitplane 6 (4 bit più significativi).
BPL6PTL	+0F6	W	A			Puntatore al bitplane 6 (15 bit meno significativi).
	0F8					
	0FA					
	0FC					
	0FE					
BPLCON0	100	W	A	D		Registro controllo bitplane 0.
BPLCON1	102	W		D		Registro controllo bitplane 1.
BPLCON2	104	W		D		Registro controllo bitplane 2.
	106					
BPL1MOD	108	W	A			Modulo dei bitplane dispari.
BPL2MOD	10A	W	A			Modulo dei bitplane pari.
	10C					
	10E					
BPL1DAT	&110	W		D		Dati del bitplane 1.
BPL2DAT	&112	W		D		Dati del bitplane 2.
BPL3DAT	&114	W		D		Dati del bitplane 3.
BPL4DAT	&116	W		D		Dati del bitplane 4.
BPL5DAT	&118	W		D		Dati del bitplane 5.
BPL6DAT	&11A	W		D		Dati del bitplane 6.
	11C					
	11E					
SPR0PTH	+120	W	A			Puntatore allo sprite 0 (4 bit più significativi).
SPR0PTL	+122	W	A			Puntatore allo sprite 0 (15 bit meno significativi).
SPR1PTH	+124	W	A			Puntatore allo sprite 1 (4 bit più significativi).
SPR1PTL	+126	W	A			Puntatore allo sprite 1 (15 bit meno significativi).
SPR2PTH	+128	W	A			Puntatore allo sprite 2 (4 bit più significativi).
SPR2PTL	+12A	W	A			Puntatore allo sprite 2 (15 bit meno significativi).
SPR3PTH	+12C	W	A			Puntatore allo sprite 3 (4 bit più significativi).
SPR3PTL	+12E	W	A			Puntatore allo sprite 3 (15 bit meno significativi).
SPR4PTH	+130	W	A			Puntatore allo sprite 4 (4 bit più significativi).
SPR4PTL	+132	W	A			Puntatore allo sprite 4 (15 bit meno significativi).
SPR5PTH	+134	W	A			Puntatore allo sprite 5 (4 bit più significativi).
SPR5PTL	+136	W	A			Puntatore allo sprite 5 (15 bit meno significativi).

SPR6PTH	+138	W	A	Puntatore allo sprite 6 (4 bit più significativi).
SPR6PTL	+13A	W	A	Puntatore allo sprite 6 (15 bit meno significativi).
SPR7PTH	+13C	W	A	Puntatore allo sprite 7 (4 bit più significativi).
SPR7PTL	+13E	W	A	Puntatore allo sprite 7 (15 bit meno significativi).
SPR0POS	%140	W	A D	Posizione iniziale dello sprite 0.
SPR0CTL	%142	W	A D	Posizione finale e controllo dello sprite 0.
SPR0DATA	%144	W	D	Registro dati A dello sprite 0.
SPR0DATB	%146	W	D	Registro dati B dello sprite 0.
SPR1POS	%148	W	A D	Posizione iniziale dello sprite 1.
SPR1CTL	%14A	W	A D	Posizione finale e controllo dello sprite 1.
SPR1DATA	%14C	W	D	Registro dati A dello sprite 1.
SPR1DATB	%14E	W	D	Registro dati B dello sprite 1.
SPR2POS	%150	W	A D	Posizione iniziale dello sprite 2.
SPR2CTL	%152	W	A D	Posizione finale e controllo dello sprite 2.
SPR2DATA	%154	W	D	Registro dati A dello sprite 2.
SPR2DATB	%156	W	D	Registro dati B dello sprite 2.
SPR3POS	%158	W	A D	Posizione iniziale dello sprite 3.
SPR3CTL	%15A	W	A D	Posizione finale e controllo dello sprite 3.
SPR3DATA	%15C	W	D	Registro dati A dello sprite 3.
SPR3DATB	%15E	W	D	Registro dati B dello sprite 3.
SPR4POS	%160	W	A D	Posizione iniziale dello sprite 4.
SPR4CTL	%162	W	A D	Posizione finale e controllo dello sprite 4.
SPR4DATA	%164	W	D	Registro dati A dello sprite 4.
SPR4DATB	%166	W	D	Registro dati B dello sprite 4.
SPR5POS	%168	W	A D	Posizione iniziale dello sprite 5.
SPR5CTL	%16A	W	A D	Posizione finale e controllo dello sprite 5.
SPR5DATA	%16C	W	D	Registro dati A dello sprite 5.
SPR5DATB	%16E	W	D	Registro dati B dello sprite 5.
SPR6POS	%170	W	A D	Posizione iniziale dello sprite 6.
SPR6CTL	%172	W	A D	Posizione finale e controllo dello sprite 6.
SPR6DATA	%174	W	D	Registro dati A dello sprite 6.
SPR6DATB	%176	W	D	Registro dati B dello sprite 6.
SPR7POS	%178	W	A D	Posizione iniziale dello sprite 7.
SPR7CTL	%17A	W	A D	Posizione finale e controllo dello sprite 7.
SPR7DATA	%17C	W	D	Registro dati A dello sprite 7.
SPR7DATB	%17E	W	D	Registro dati B dello sprite 7.
COLOR00	180	W	D	Registro di colore 00.
COLOR01	182	W	D	Registro di colore 01.
COLOR02	184	W	D	Registro di colore 02.
COLOR03	186	W	D	Registro di colore 03.
COLOR04	188	W	D	Registro di colore 04.
COLOR05	18A	W	D	Registro di colore 05.
COLOR06	18C	W	D	Registro di colore 06.
COLOR07	18E	W	D	Registro di colore 07.
COLOR08	190	W	D	Registro di colore 08.
COLOR09	192	W	D	Registro di colore 09.
COLOR10	194	W	D	Registro di colore 10.
COLOR11	196	W	D	Registro di colore 11.
COLOR12	198	W	D	Registro di colore 12.
COLOR13	19A	W	D	Registro di colore 13.
COLOR14	19C	W	D	Registro di colore 14.

COLOR15	19E	W	D	Registro di colore 15.
COLOR16	1A0	W	D	Registro di colore 16.
COLOR17	1A2	W	D	Registro di colore 17.
COLOR18	1A4	W	D	Registro di colore 18.
COLOR19	1A6	W	D	Registro di colore 19.
COLOR20	1A8	W	D	Registro di colore 20.
COLOR21	1AA	W	D	Registro di colore 21.
COLOR22	1AC	W	D	Registro di colore 22.
COLOR23	1AE	W	D	Registro di colore 23.
COLOR24	1B0	W	D	Registro di colore 24.
COLOR25	1B2	W	D	Registro di colore 25.
COLOR26	1B4	W	D	Registro di colore 26.
COLOR27	1B6	W	D	Registro di colore 27.
COLOR28	1B8	W	D	Registro di colore 28.
COLOR29	1BA	W	D	Registro di colore 29.
COLOR30	1BC	W	D	Registro di colore 30.
COLOR31	1BE	W	D	Registro di colore 31.

Sono riservati per uso futuro i registri:

	1CX	
	1DX	
	1EX	
	1FX	
NO-OP	1FE	Registro nullo

D MAPPA DI MEMORIA DEL SISTEMA

Una mappa di memoria che mostri come vengono utilizzate le varie sezioni di RAM da parte delle risorse del sistema è resa impossibile dalla stessa architettura dell'Amiga. Tutta la memoria viene allocata dinamicamente, e le locazioni possono variare a seconda della versione del sistema, del modello di Amiga, e addirittura da un boot all'altro. Per trovare le locazioni corrispondenti alle varie strutture dati usate dal sistema, è necessario ricorrere alle normali procedure di accesso: iniziare cioè dall'indirizzo della libreria Exec, contenuto nella locazione 4 (l'unico indirizzo assoluto del sistema). Tutti i programmi devono essere scritti in modo da poter essere caricati in qualunque area di memoria dal loader del sistema.

Quello che segue è uno schema generale delle aree di memoria contenute nella generazione attuale dei computer Amiga.

INDIRIZZI	**NOTE**
000000-03FFFF	256K di memoria chip.
040000-07FFFF	256K di memoria chip (opzionali sull'A1000).
080000-0FFFFF	512K di memoria chip estesa (1MB totale).
100000-1FFFFF	Riservati. Non utilizzare.
200000-9FFFFF	8MB di memoria fast AUTOCONFIG.
A00000-BEFFFF	Riservati. Non utilizzare.
BFD000-BFDF00	8520-B (accesso riservato a indirizzi pari).
BFE001-BFEF01	8520-A (accesso riservato a indirizzi dispari).

ASSEGNAZIONE DEI PIN DEL CHIP DENISE

PIN	NOME	FUNZIONE	DEFINIZIONE
01-07	D6-D0	Linee del bus dati 6-0	I/O
08	M1H	Impulso orizzontale mouse 1	I
09	M0H	Impulso orizzontale mouse 0	I
10-17	RGA8-RGA1	Bus indirizzi dei registri 8-1	I
18	BURST*	Impulso del colore	O
19	VCC	+5 Volt	I
20-23	R0-R3	Bit del segnale del rosso 0-3	O
24-27	B0-B3	Bit del segnale del blu 0-3	O
28-31	G0-G3	Bit del segnale del verde 0-3	O
32	N/C	Non collegato	-
33	ZD*	Indicatore di background	O
34	N/C	Non collegato	-
35	7M	7,09379 MHz	I
36	CCK	Clock di colore	I
37	VSS	Terra	I
38	M0V	Impulso verticale mouse 0	I
39	M1V	Impulso verticale mouse 1	I
40-48	D15-D7	Linee del bus dati 15-7	I/O

ASSEGNAZIONE DEI PIN DEL CHIP PAULA

PIN	NOME	FUNZIONE	DEFINIZIONE
01-07	D8-D2	Linee del bus dati 8-2	I/O
08	VSS	Terra	I
09-10	D1-D0	Linee del bus dati 1-0	I/O
11	RES*	Reset del sistema	I
12	DMAL	Linea di richiesta del DMA	O
13-15	IPL0*-IPL2*	Linee di richiesta di interrupt 0-2	O
16	INT2*	Interrupt di livello 2	I
17	INT3*	Interrupt di livello 3	I
18	INT6*	Interrupt di livello 6	I
19-26	RGA8-RGA1	Bus indirizzi dei registri 8-1	I
27	VCC	+5 Volt	I
28	CCK	Clock di colore	I
29	CCKQ	Ritardo sul clock di colore	I
30	AUDB	Canale audio destro	O
31	AUDA	Canale audio sinistro	O
32	POT0X	Pot 0 X	I/O
33	POT0Y	Pot 0 Y	I/O
34	VSSANA	Terra, analogico	I
35	POT1X	Pot 1 X	I/O
36	POT1Y	Pot 1 Y	I/O
37	DKRD*	Dati in lettura dal disco	I
38	DKWD*	Dati in scrittura al disco	O
39	DKWE	Protezione in scrittura del disco	O

40	TXD	Dati trasmessi dalla porta seriale	O
41	RXD	Dati ricevuti dalla porta seriale	I
42-48	D15-D9	Linee del bus dati 15-9	I/O

ASSEGNAZIONE DEI PIN DEL CHIP FAT AGNUS

PIN	NOME	FUNZIONE	DEFINIZIONE
01-14	RD15-RD2	Linee del bus dei registri 15-2	I/O
15	INT3*	Interrupt di blit concluso	O
16	DMAL	Richiesta di DMA audio/dischi	I
17	RD1	Linea 1 del bus dei registri	I/O
18	RST*	Reset	I
19	BLS*	Rallentamento del Blitter	I
20	DBR*	Richiesta del bus dati	O
21	RRW	Lettura/scrittura su DRAM	O
22	PRW	Lettura/scrittura sul processore	I
23	RGEN*	Abilitazione RG	I
24	AS*	Strobe degli indirizzi	I
25	RAMEN*	Abilitazione RAM	I
26-33	RGA8-RGA1	Bus indirizzi destinazione 8-1	O
34	28MHZ	Clock principale	I
35	XCLK	Secondo clock principale	I
36	XCLKEN*	Abilitazione del clock principale	I
37	CDAC*	Clock a 7 MHz invertito	O
38	7MHZ	28 MHz diviso 4	O
39	CCKQ	Ritardo sul clock di colore	O
40	CCK	Clock di colore	O
41	TEST	Test di accesso ai registri	I
43-51	MA0-MA8	Linee del bus di output 0-8	O
52	LDS*	Strobe dati inferiore	I
53	UDS*	Strobe dati superiore	I
54	CASL*	Indirizzo colonna strobe inferiore	O
55	CASU*	Indirizzo colonna strobe superiore	O
56	RAS1*	Indirizzo riga strobe 1	O
57	RAS0*	Indirizzo riga strobe 0	O
59-77	A19-A1	Linee del bus indirizzi 19-1	I
78	LP*	Penna ottica	I
79	VSY*	Sincronismo verticale	I/O
80	CSY*	Sincronismo videocomposito	O
81	HSY*	Sincronismo orizzontale	I/O
84	RD0	Linea 0 del bus registri	I/O

C ELENCO DEI PIN DEI CHIP CUSTOM

NOTA: * indica un segnale attivo basso.

ASSEGNAZIONE DEI PIN DEL CHIP AGNUS

PIN	NOME	FUNZIONE	DEFINIZIONE
01-09	D8-D0	Linee del bus dati 8-0	I/O
10	VCC	+5 Volt	I
11	RES*	Reset del sistema	I
12	INT3*	Interrupt di livello 3	O
13	DMAL	Linea di richiesta DMA	I
14	BLS*	Rallentamento del Blitter	I
15	DBR*	Richiesta del bus dati	O
16	ARW*	Scrittura da Agnus a RAM	O
17-24	RGA8-RGA1	Bus indirizzi dei registri 8-1	I/O
25	CCK	Clock di colore	I
26	CCKQ	Ritardo sul clock di colore	I
27	VSS	Terra	I
28-36	DRA0-DRA8	Bus indirizzi DRAM 8-1	O
37	LP*	Input della penna ottica	I
38	VSY*	Sincronismo verticale	I/O
39	CSY*	Sincronismo composito	O
40	HSY*	Sincronismo orizzontale	I/O
41	VSS	Terra	I
42-48	D15-D9	Linee del bus dati 15-9	I/O

C00000-DFEFFF	Riservati. Non utilizzare.
C00000-D7FFFF	Espansione di memoria interna.
D80000-DBFFFF	Riservati. Non utilizzare.
DC0000-DCFFFF	Orologio interno in tempo reale.
DFF000-DFFFFF	Registri dei chip custom.
E00000-E7FFFF	Riservati. Non utilizzare.
E80000-E8FFFF	Spazio di AUTOCONFIG. Le schede appaiono a questo indirizzo finché il sistema non le riloca al loro indirizzo finale.
E90000-EFFFFF	Spazio di AUTOCONFIG secondario. Utilizzato generalmente da schede I/O da 64K.
F00000-FBFFFF	Riservati. Non utilizzare.
FC0000-FFFFFF	256K di ROM di sistema.

E INTERFACCE

Questa appendice è divisa in tre parti distinte, relative al modo con cui l'Amiga dialoga con il mondo esterno.

La prima parte descrive i pin dei connettori esterni. Non fornisce, tuttavia, informazioni sulle temporizzazioni e sull'uso da parte del sistema.

La seconda parte descrive brevemente il significato dei pin la cui funzione potrebbe non essere evidente.

La terza parte contiene un elenco di collegamenti relativi ad alcuni connettori interni, come quello del drive.

Il software sistema è strutturato in maniera tale da occuparsi senza problemi dell'impostazione di particolari segnali, anche se i collegamenti fisici variano a seconda dei diversi modelli di Amiga. In altre parole, possedendo una versione del software sistema che corrisponde al livello di revisione della macchina (una condizione normale), non è necessario sapere esattamente a quale porta 'è collegato un bit per richiederne l'impostazione. I programmatori, pertanto, dovrebbero basarsi sulla documentazione relativa al sistema operativo, evitando di collegarsi direttamente alle porte.

In un ambiente operativo multitasking come quello dell'Amiga, diversi task possono entrare in competizione per l'uso delle risorse del sistema. I programmi devono perciò seguire le regole stabilite per il corretto accesso all'hardware, in maniera da assicurare la piena compatibilità con il sistema operativo.

Parte 1

Porta RS232 e MIDI

PIN	RS232	A1000	A500/ A2000	PC CBM	HAYES	DESCRIZIONE
1	GND	GND	GND	GND	GND	TERRA
2	TXD	TXD	TXD	TXD	TXD	DATI IN USCITA
3	RXD	RXD	RXD	RXD	RXD	DATI IN INGRESSO
4	RTS	RTS	RTS	RTS	–	PRONTO A TRASMETTERE
5	CTS	CTS	CTS	CTS	CTS	PRONTO A RICEVERE
6	DSR	DSR	DSR	DSR	DSR	DATI PRONTI
7	GND	GND	GND	GND	GND	TERRA
8	CD	CD	CD	DCD	DCD	CARRIER
9	–	–	+12V	+12V	–	+12 VOLT
10	–	–	–12V	–12V	–	–12 VOLT
11	–	–	AUDO	–	–	OUTPUT AUDIO
12	S.SD	–	–	–	SI	INDICATORE DI VELOCITÀ
13	S.CTS	–	–	–	–	
14	S.TXD	–5V	–	–	–	–5 VOLT
15	TXC	AUDO	–	–	–	OUTPUT AUDIO
16	S.RXD	AUDI	–	–	–	INPUT AUDIO
17	RXC	EB	–	–	–	CLOCK 7 MHZ
18	–	INT2*	AUDI	–	–	INTERRUPT
19	S.RTS	–	–	–	–	
20	DTR	DTR	DTR	DTR	DTR	TERMINALE PRONTO
21	SQD	+5V	–	–	–	+5 VOLT
22	RI	–	RI	RI	RI	INDICATORE DI RING
23	SS	+12V	–	–	–	+12 VOLT
24	TXC1	C2*	–	–	–	CLOCK 3,5 MHZ
25	–	RESB*	–	–	–	RESET

Porta parallela (Centronics)

PIN	A1000	A500/A2000	PC CBM
1	DRDY*	STROBE*	STROBE*
2	Data 0	Data 0	Data 0
3	Data 1	Data 1	Data 1
4	Data 2	Data 2	Data 2
5	Data 3	Data 3	Data 3
6	Data 4	Data 4	Data 4
7	Data 5	Data 5	Data 5
8	Data 6	Data 6	Data 6
9	Data 7	Data 7	Data 7
10	ACK*	ACK*	ACK*
11	BUSY (data)	BUSY	BUSY
12	POUT (clk)	POUT	POUT
13	SEL	SEL	SEL

14	GND	+5V	AUTOFDXT
15	GND	NC	ERROR*
16	GND	RESET*	INIT*
17	GND	GND	SLCT IN*
18-22	GND	GND	GND
23	+5V	GND	GND
24	NC	GND	GND
25	RESET*	GND	GND

Tastiera - RJ11

PIN	A1000	A2000
1	+5V	KCLK
2	CLOCK	KDAT
3	DATA	NC
4	GND	GND
5	-	+5V

Non applicabile all'A500.

Video - DB23 maschio

Valido per A500, A1000, A2000 salvo indicazioni contrarie.

1	XCLK*	
2	XCLKEN*	
3	RED	
4	GREEN	
5	BLUE	
6	DI	
7	DB	
8	DG	
9	DR	
10	CSYNC*	
11	HSYNC*	
12	VSYNC*	
13	GNDRTN (Ritorno di XCLKEN*)	
14	ZD*	
15	C1*	
16	GND	
17	GND	
18	GND	
19	GND	
20	GND	
21	A1000/A2000	-5 VOLT
	A500	-12 VOLT
22	+12 VOLT	
23	+5 VOLT	

RF Monitor – 8 pin DIN (J2)
Solo A1000.

1	NC
2	GND
3	AUDIO SINISTRO
4	VIDEOCOMPOSITO
5	GND
6	NC
7	+12 VOLT
8	AUDIO DESTRO

Drive esterno – DB23 femmina
Valido per A500, A1000, A2000 salvo indicazioni contrarie.

1	RDY*
2	DKRD*
3	GND
4	GND
5	GND
6	GND
7	GND
8	MTRXD*
9	SEL2B* (A2000 SEL3B* (1))
10	DRESB*
11	CHNG*
12	+5V
13	SIDEB*
14	WPRO*
15	TK0*
16	DKWEB*
17	DKWDB*
18	STEPB*
19	DIRB
20	SEL3B* (non utilizzato su A2000 (1))
21	SEL1B* (A2000 SEL2B* (1))
22	INDEX*
23	+12V

(1) Sull'A2000 SEL1B* non corrisponde al drive 1 ma al primo drive esterno. Non tutte le possibili linee di selezione sono necessariamente installate.

RAMEX - 60 pin EDGE (.156) (P1)
Solo A1000.

1	GND	A	GND
2	D15	B	D14
3	+5V	C	+5V
4	D12	D	D13
5	GND	E	GND
6	D11	F	D10
7	+5V	H	+5V
8	D8	J	D9
9	GND	K	GND
10	D7	L	D6
11	+5V	M	+5V
12	D4	N	D5
13	GND	P	GND
14	D3	R	D2
15	+5V	S	+5V
16	D0	T	D1
17	GND	U	GND
18	DRA4	V	DRA3
19	DRA5	W	DRA2
20	DRA6	X	DRA1
21	DRA7	Y	DRA0
22	GND	Z	GND
23	RAS*	AA	RRW*
24	GND	BB	GND
25	GND	CC	GND
26	CASU0*	DD	CASU1*
27	GND	EE	GND
28	CASL0*	FF	CASL1*
29	+5V	HH	+5V
30	+5V	JJ	+5V

Espansione – 86 pin EDGE (.1) (P2)

PIN	A500	A1000	A2000	A2000B	FUNZIONE
1	x	x	x	x	Terra
2	x	x	x	x	Terra
3	x	x	x	x	Terra
4	x	x	x	x	Terra
5	x	x	x	x	+5V
6	x	x	x	x	+5V
7	x	x	x	x	Nessuna connessione
8	x	x	x	x	–5V
9	x	x			Nessuna connessione
			x	x	28 MHz
10	x	x	x	x	+12V
11	x	x	x		Nessuna connessione
				x	/COPCFG (CONFIG OUT)
12	x	x	x	x	CONFIG IN (Terra)
13	x	x	x	x	Terra
14	x	x	x	x	/C3 clock
15	x	x	x	x	CDAC clock
16	x	x	x	x	/C1 clock
17	x	x	x	x	/OVR
18	x	x	x	x	RDY
19	x	x	x	x	/INT2
20		x			/PALOPE
	x		x		Nessuna connessione
				x	/BOSS
21	x	x	x	x	A5
22	x	x	x	x	/INT6
23	x	x	x	x	A6
24	x	x	x	x	A4
25	x	x	x	x	Terra
26	x	x	x	x	A3
27	x	x	x	x	A2
28	x	x	x	x	A7
29	x	x	x	x	A1
30	x	x	x	x	A8
31	x	x	x	x	FC0
32	x	x	x	x	A9
33	x	x	x	x	FC1
34	x	x	x	x	A10
35	x	x	x	x	FC2
36	x	x	x	x	A11
37	x	x	x	x	Terra
38	x	x	x	x	A12
39	x	x	x	x	A13
40	x	x	x	x	/IPL0
41	x	x	x	x	A14
42	x	x	x	x	/IPL1
43	x	x	x	x	A15

44	x	x	x	x	/IPL2
45	x	x	x	x	A16
46	x	x	x	x	BEER*
47	x	x	x	x	A17
48	x	x	x	x	/VPA
49	x	x	x	x	Terra
50	x	x	x	x	E clock
51	x	x	x	x	/VMA
52	x	x	x	x	A18
53	x	x	x	x	RST
54	x	x	x	x	A19
55	x	x	x	x	/HLT
56	x	x	x	x	A20
57	x	x	x	x	A22
58	x	x	x	x	A21
59	x	x	x	x	A23
60	x	x	x		/BR
				x	/CBR
61	x	x	x	x	Terra
62	x	x	x	x	/BGACK
63	x	x	x	x	D15
64	x	x	x		/BG
				x	/CBG
65	x	x	x	x	D14
66	x	x	x	x	/DTACK
67	x	x	x	x	D13
68	x	x	x	x	R/W
69	x	x	x	x	D12
70	x	x	x	x	/LDS
71	x	x	x	x	D11
72	x	x	x	x	/UDS
73	x	x	x	x	Terra
74	x	x	x	x	/AS
75	x	x	x	x	D0
76	x	x	x	x	D10
77	x	x	x	x	D1
78	x	x	x	x	D9
79	x	x	x	x	D2
80	x	x	x	x	D8
81	x	x	x	x	D3
82	x	x	x	x	D7
83	x	x	x	x	D4
84	x	x	x	x	D6
85	x	x	x	x	Terra
86	x	x	x	x	D5

Joystick – DB9 maschio

PIN	JOYSTICK	MOUSE
1	FORWARD*	MOUSE V
2	BACK*	MOUSE H
3	LEFT*	MOUSE VQ
4	RIGHT*	MOUSE HQ
5	POT X	Pulsante 3
6	FIRE*	Pulsante 1
7	+5V	+5V
8	GND	GND
9	POT Y	Pulsante 2

Parte 2

Specifiche di connessione dell'interfaccia parallela

Il connettore a 25 pin (DB25P maschio per l'Amiga 1000, femmina per Amiga 500/2000 e IBM compatibili) viene generalmente usato come interfaccia per stampanti parallele. In questo caso il flusso di dati va dal computer alla stampante. L'interfaccia è però capace di trasferimenti bidirezionali. L'uso è simile a quello dell'interfaccia Centronics, ma in alcuni casi l'assegnazione dei pin e le caratteristiche di funzionamento sono piuttosto diverse. I nomi attribuiti ai segnali corrispondono in linea di massima a quelli usati altrove in questa appendice.

Assegnazione dei pin del connettore parallelo

NOME	DIREZIONE	NOTE
DRDY*	O	Segnale di dati pronti per essere inviati al dispositivo esterno (modo output). Viene usato in congiunzione con ACK* (pin 10) per un handshake asincrono a due linee. Funziona come linea dati in ingresso in modo input (in maniera simile ad ACK* in modo output). Vedere anche i diagrammi di temporizzazione.
D0	I/O	+
D1	I/O	|
D2	I/O	|
D3	I/O	| D0-D7 costituiscono un bus dati bidirezionale a 8 bit per la comunicazione con dispositivi esterni.
D4	I/O	|
D5	I/O	|
D6	I/O	|
D7	I/O	+
ACK*	I	Riconoscimento di dati provenienti dal dispositivo esterno in modo output. Viene usato in congiunzione con DRDY* (pin 1) per un handshake asincrono a due linee. Funziona come segnale di dati pronti dal dispositivo esterno in modo input (in maniera simile a DRDY* in modo output). Vedere anche i diagrammi di temporizzazio-

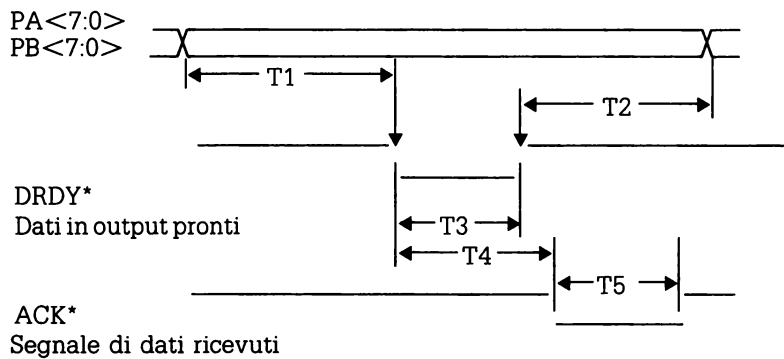
ne. L'8520 può essere programmato in modo da generare un interrupt di livello 2 ogni volta che diventa attiva la linea di input ACK*.

BUSY I/O — Pin generico di I/O collegato anche a un pin di dati seriale (clock seriale al pin 12). Nominalmente usato come indicatore del riempimento del buffer dati della stampante.

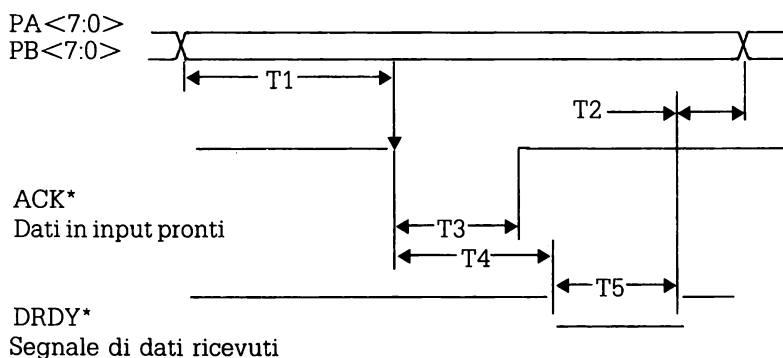
POUT I/O — Pin generico di I/O collegato a un pin di clock seriale (dati seriali al pin 11). Nominalmente usato per indicare la condizione di fine carta.

SEL I/O — Generico pin di I/O. Nominalmente usato per selezionare l'output di dati dal dispositivo esterno all'Amiga. Su A500/A2000 condivide il segnale di ring tipico della porta RS232.

RESET* O — Indicatore del reset del sistema.

Temporizzazione di un ciclo di output dell'interfaccia parallela

	Microsecondi			
	Min	**Tip**	**Max**	
T1:	4,3	-x-	5,3	Intervallo di preparazione dati.
T2:	nsp	-x-	scp	Durata dei dati sul bus.
T3:	nsp	1,4	nsp	Intervallo di dati pronti.
T4:	0	-x-	scp	Intervallo di attesa riconoscimento dati.
T5:	nsp	-x-	scp	Durata del riconoscimento dati.

nsp = non specificato
scp = sotto il controllo del programma

Temporizzazione di un ciclo di input dell'interfaccia parallela

	Microsecondi Min	Tip	Max	
T1:	0	-x-	scp	Intervallo di preparazione dati in input.
T2:	nsp	-x-	scp	Durata dei dati sul bus.
T3:	nsp	-x-	scp	Intervallo di dati pronti.
T4:	scp	-x-	scp	Intervallo fra dati pronti e riconoscimento dati.
T5:	nsp	1,4	nsp	Durata del riconoscimento dati.

nsp = non specificato
scp = sotto il controllo del programma

Specifiche di connessione dell'interfaccia seriale

Questo connettore a 25 pin (DB25S femmina) viene usato come interfaccia per segnali RS232C standard. I nomi attribuiti ai vari segnali corrispondono in linea di massima a quelli usati in altre parti di questa appendice.

ATTENZIONE: pin sul connettore RS232 diversi da quelli descritti di seguito potrebbero essere collegati a linee di alimentazione o ad altri segnali non standard. Si utilizzino cavi appropriati in cui siano collegati solo i segnali realmente utilizzati e si evitino generici cavi di connessione a 25 pin.

Assegnazione dei pin dell'interfaccia seriale (J6)

NOME	DIR.	STD	NOTE
FGND		S	Terra fisica (non collegare a terra logica).
TXD	O	S	Trasmissione dati.
RXD	I	S	Ricezione dati.
RTS	O	S	Pronto a trasmettere
CTS	I	S	Pronto a ricevere
DSR	I	S	Dati pronti.

CD	I	S	Carrier.
−5V		N	max 50 mA.
AUDO	O	N	Output del canale sinistro (canali 0, 3). Pensato per l'invio dell'audio al modem.
AUDI	I	N	Input audio sul canale destro (canali 1, 2). Pensato per la ricezione audio dal modem. Questo input viene fuso con l'output analogico proveniente dai canali 1 e 2 e non viene digitalizzato o utilizzato dal computer in nessun modo.
DTR	O	S	Terminale pronto.
RI	I	S	Indicatore di ring (solo su A500/A2000) condiviso con il segnale di selezione della stampante.
RESB*	O	N	Reset dell'Amiga.

(Vedere la Parte 1 per i numeri dei pin corrispondenti).

Temporizzazione dell'interfaccia seriale

La massima frequenza operativa è di 19,2 KHz. Per maggiori specifiche sull'installazione e sull'uso si consultino i documenti dello standard EIA relativi all'interfaccia RS232C. L'interfaccia può raggiungere una frequenza di 31,25 KHz tramite un adattatore MIDI.

I segnali di controllo del modem (CTS, RTS, DTR, DSR, CD) sono sotto completo controllo software e non hanno alcun effetto sull'hardware essendo completamente asincroni da TXD e RXD.

Caratteristiche elettriche dell'interfaccia seriale

OUTPUT	MIN	TIP	MAX		
Vo (−):	−13,2	-x-	−2,5	V	Intervallo della tensione negativa.
Vo (+):	8,0	-x-	13,2	V	Intervallo della tensione positiva.
Io:	-x-	-x-	10,0	ma	Corrente in uscita.

INPUT	MIN	TIP	MAX		
Vi (+):	3,0	-x-	25,0	V	Intervallo della tensione positiva.
Vi (−):	−25,0	-x-	0,5	V	Intervallo della tensione negativa.
Vis:	-x-	1,0	-x-	V	Tensione d'isteresi.
Ii:	0,3	-x-	10,0	ma	Corrente in ingresso.

Specifiche di connessione delle porte di controllo

I due connettori a 9 pin di tipo D maschi vengono usati come interfaccia con quattro tipi di dispositivi:

1. Mouse o trackball (massimo 3 pulsanti).
2. Joystick digitale (massimo 2 pulsanti).
3. Input proporzionale (paddle o joystick); (massimo 2 pulsanti).
4. Penna ottica, con pulsante di attivazione.

L'assegnazione dei vari pin è già stata discussa nei paragrafi relativi ai dispositivi menzionati. I nomi attribuiti ai vari segnali rimangono in linea di massima immutati.

J11 è il connettore relativo alla porta di destra (JOY1DAT, POT1DAT).
J12 è il connettore relativo alla porta di sinistra (JOY0DAT, POT0DAT).

Interfaccia con gli input di quadratura di mouse/trackball

Un mouse o una trackball sono dispositivi che trasformano un movimento su un piano in una serie di impulsi. Per conservare le informazioni relative alla direzione e all'entità dello spostamento, vengono utilizzate tecniche dette "di quadratura". I registri JOY0DAT e JOY1DAT diventano registri contatore con i dati relativi agli spostamenti verticali (y) nel byte più significativo e quelli relativi agli spostamenti orizzontali (x) in quello meno significativo. Il movimento dà origine alle seguenti azioni:

Verso l'alto	decremento y
Verso il basso	incremento y
Verso destra	incremento x
Verso sinistra	decremento y

Per determinare l'entità del movimento, si deve leggere due volte JOYxDAT e confrontare le differenze fra le due coppie di valori x e y (attenzione, si tratta di aritmetica in modulo 128). Si noti che se uno dei due contatori cambia di una quantità superiore a 127, distanza e direzione diventano indeterminate. Ecco la relazione fra intervallo di campionamento e massima velocità verificabile senza incertezze:

$$\text{Velocità} < \text{Distanza massima/Intervallo di campionamento}$$

$$\text{Velocità} < \sqrt{(\Delta X)^2 + (\Delta Y)^2}/\text{Intervallo di campionamento}$$

Per un mouse che dà origine a circa 79 incrementi/decrementi per centimetro, controllati a ogni intervallo di vertical blanking, la massima velocità (orizzontale o verticale) risulta:

$$\text{Velocità} < (128 * 1/79)/0{,}02 = 81 \text{ cm/sec}$$

che dovrebbe essere sufficiente nella maggior parte dei casi.

NOTA: il software dell'Amiga è disegnato in maniera tale da aggiornare la posizione del mouse a ogni intervallo di vertical blanking. I contatori sono pertanto sempre validi e possono essere letti in qualunque momento.

Connessioni per gli input di quadratura del mouse

PIN	NOME	DESCRIZIONE	NOTE
1	V	Impulso verticale	JOYxDAT<15:8>
2	H	Impulso orizzontale	JOYxDAT<15:8>
3	VQ	Impulso di quadratura verticale	JOYxDAT<15:8>
4	HQ	Impulso di quadratura orizzontale	JOYxDAT<15:8>
5	UBUT*	Pulsante non utilizzato dal mouse	Vedere input proporzionale
6	LBUT*	Pulsante sinistro del mouse	Vedere pulsante Fire
7	+5V	+5 Volt, corrente limitata	
8	Terra		
9	RBUT*	Pulsante destro del mouse	Vedere input proporzionale

Interfacciamento con joystick digitali

Un joystick è un dispositivo che presenta quattro interruttori normalmente in posizione "aperta", disposti a novanta gradi l'uno dall'altro. I registri JOYxDAT hanno il seguente significato:

Avanti	bit 9 XOR bit 8
Sinistra	bit 9
Indietro	bit 1 XOR bit 0
Destra	bit 1

I dati sono codificati in questo modo per facilitare le operazioni di lettura con mouse o trackball.

NOTA: le direzioni di destra e di sinistra sono disegnate in modo da poter essere utilizzate anche come pulsanti Fire, rispettivamente destro e sinistro, con input di tipo proporzionale. In questo caso, le direzioni avanti e indietro non vengono utilizzate.

I registri JOYxDAT sono sempre validi e possono essere letti in qualunque momento.

Connessioni per input da joystick digitale

PIN	NOME	DESCRIZIONE	NOTE
1	FORWARD*	Avanti	JOYxDAT<9 XOR 8>
2	BACK*	Indietro	JOYxDAT<1 XOR 0>
3	LEFT*	Sinistra	JOYxDAT<9>
4	RIGHT*	Destra	JOYxDAT<1>
5	Non utilizzato		
6	FIRE*	Pulsante Fire	Vedere pulsanti Fire
7	+5V	125 mA max, 200 mA di sovracorrente	 Totale per entrambe le porte
8	Terra		
9	Non utilizzato		

Interfacciamento con i pulsanti Fire

I pulsanti Fire sono interruttori normalmente aperti, collegati con il registro PRA0 del chip 8520 CIAA:

bit 7 di PRA0 = FIRE*, porta di sinistra
bit 6 di PRA0 = FIRE*, porta di destra

Prima di leggere questi registri si devono azzerare i corrispondenti bit del registro di direzione per definire il modo input:

DDRA0<7:6> = 0

NOTA: non modificare lo stato degli altri bit di DDRA0 (si raccomanda l'uso delle routine del sistema operativo).

I pulsanti Fire sono sempre validi e possono essere letti in qualunque momento.

Connessioni per input da pulsanti Fire

PIN	NOME	DESCRIZIONE
1	-x-	
2	-x-	
3	-x-	
4	-x-	
5	-x-	
6	FIRE*	Pulsante sinistro del mouse o pulsante Fire
7	-x-	
8	Terra	
9	-x-	

Interfacciamento con dispositivi proporzionali

Sono previsti elementi a resistenza variabile (potenziometri) lineari fino a un massimo di 528 KΩ (valore raccomandato 470KΩ +/- 10%). I registri POTxDAT contengono i valori corrispondenti alla y nel byte più significativo e quelli corrispondenti alla x nel byte meno significativo. Vi è una proporzionalità diretta tra i valori contenuti nel registro e la resistenza esterna. La conversione si esegue nel modo seguente:

1. Per le prime 8 (7 in NTSC) linee orizzontali dello schermo, i condensatori interni vengono scaricati e i contatori contenuti in POTxDAT vengono azzerati.
   Per la rimanente parte del quadro video, i condensatori si ricaricano attraverso la resistenza contenuta nel dispositivo esterno.

2. La carica in aumento viene continuamente confrontata con un valore fisso di riferimento mentre i contatori tengono conto del numero di linee orizzontali percorso dal termine dell'intervallo di reset.

3. Quando per un certo canale la carica supera il valore di riferimento, il valore del contatore viene trasferito nel registro POTxDAT corrispondente a quel canale.

4. Durante il successivo intervallo di vertical blanking, il software esamina i valori contenuti nei registri POTxDAT e li interpreta in termini di posizione del joystick.

NOTA: i contatori POTX e POTY sono anche designati, rispettivamente, come pulsante destro del mouse ed eventuale terzo pulsante. Un pulsante premuto corrisponde a una bassa resistenza, un pulsante non premuto a una resistenza alta. I valori sono disponibili anche nei registri POTGO e POTINP. Si raccomanda l'uso delle routine del sistema operativo per la lettura di questi valori, allo scopo di garantire la compatibilità con versioni future dell'hardware.

È importante rendersi conto che i dispositivi proporzionali non forniscono esattamente valori assoluti di posizione. Spetta al software controllare la calibrazione, limitare l'intervallo dei valori accettabili e tradurli in valori medi che un programma applicativo possa utilizzare.

In genere i registri POTxDAT vengono letti durante l'intervallo di vertical blanking, ma possono essere utilizzati anche prima.

Connessioni per input proporzionale

PIN	NOME	DESCRIZIONE	NOTE
1	XBUT	Pulsante supplementare	
2	Non utilizzato		
3	LBUT*	Pulsante sinistro	Vedere joystick digitale
4	RBUT*	Pulsante destro	Vedere joystick digitale
5	POTX	Input proporzionale X	POTxDAT<7:0>, POTGO, POTINP
6	Non utilizzato		
7	+5V	125 mA max, 200 mA di sovracorrente	
8	Terra		
9	POTY	Input proporzionale Y	POTxDAT<7:0>, POTGO, POTINP

Interfacciamento con una penna ottica

Una penna ottica è un dispositivo optoelettronico la cui parte sensibile alla luce viene tenuta sullo schermo video. Quando il pennello elettronico giunge in prossimità della penna, questa genera un impulso che può essere utilizzato per rilevare la posizione del pennello elettronico. L'evento è rilevabile anche se nel sistema non esiste un bit che lo segnali (si veda il Capitolo 8).

La posizione della penna ottica viene generalmente letta durante l'intervallo di vertical blanking, ma *potrebbe* essere disponibile anche prima.

Connessioni per il collegamento della penna ottica

PIN	NOME	DESCRIZIONE	NOTE
1	Non utilizzato		
2	Non utilizzato		
3	Non utilizzato		
4	Non utilizzato		
5	LPENPR*	Penna premuta sullo schermo	Vedere input proporzionali
6	LPENTG*	Aggancio del pennello elettronico	VPOSR, VHPOSR
7	+5V	125 mA max, 200 mA di sovracorrente	
8	Terra		
9	Non utilizzato		

Parte 3

CONNETTORI INTERNI

Disk drive interno – 34 pin (J10)

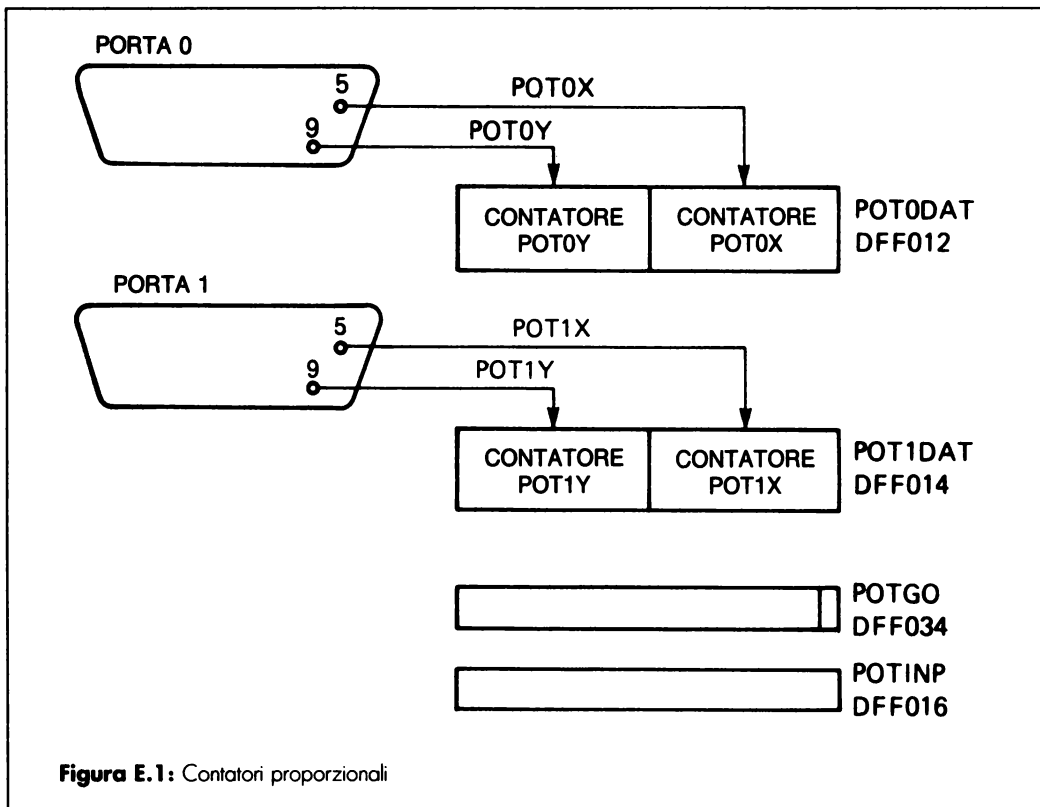
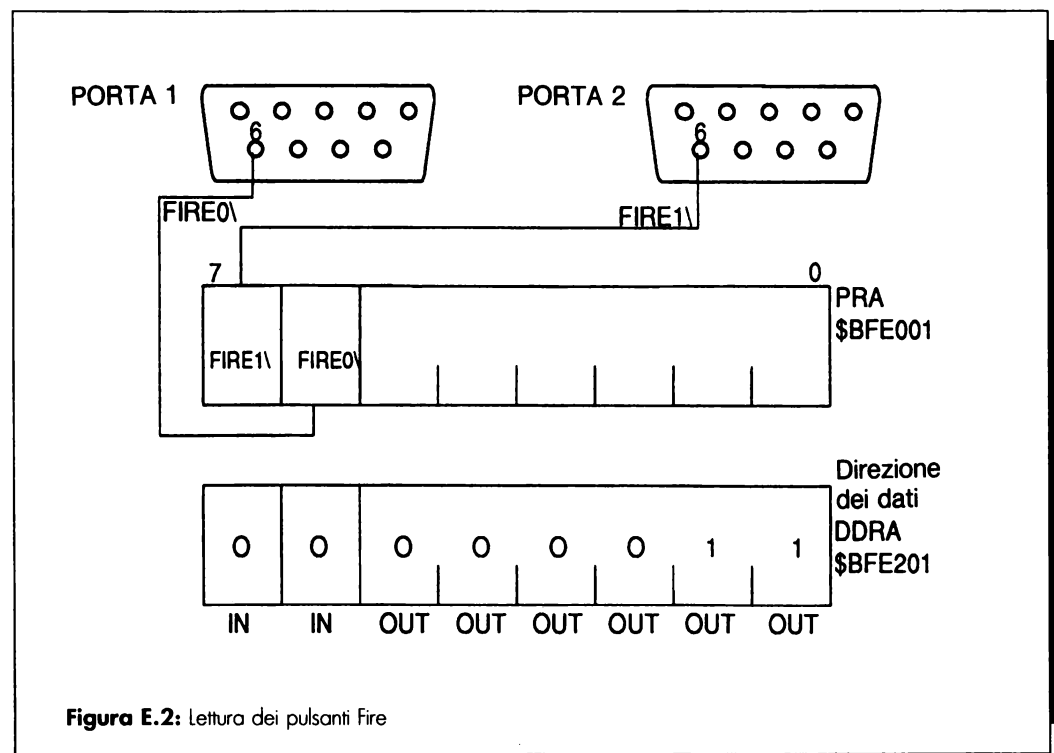
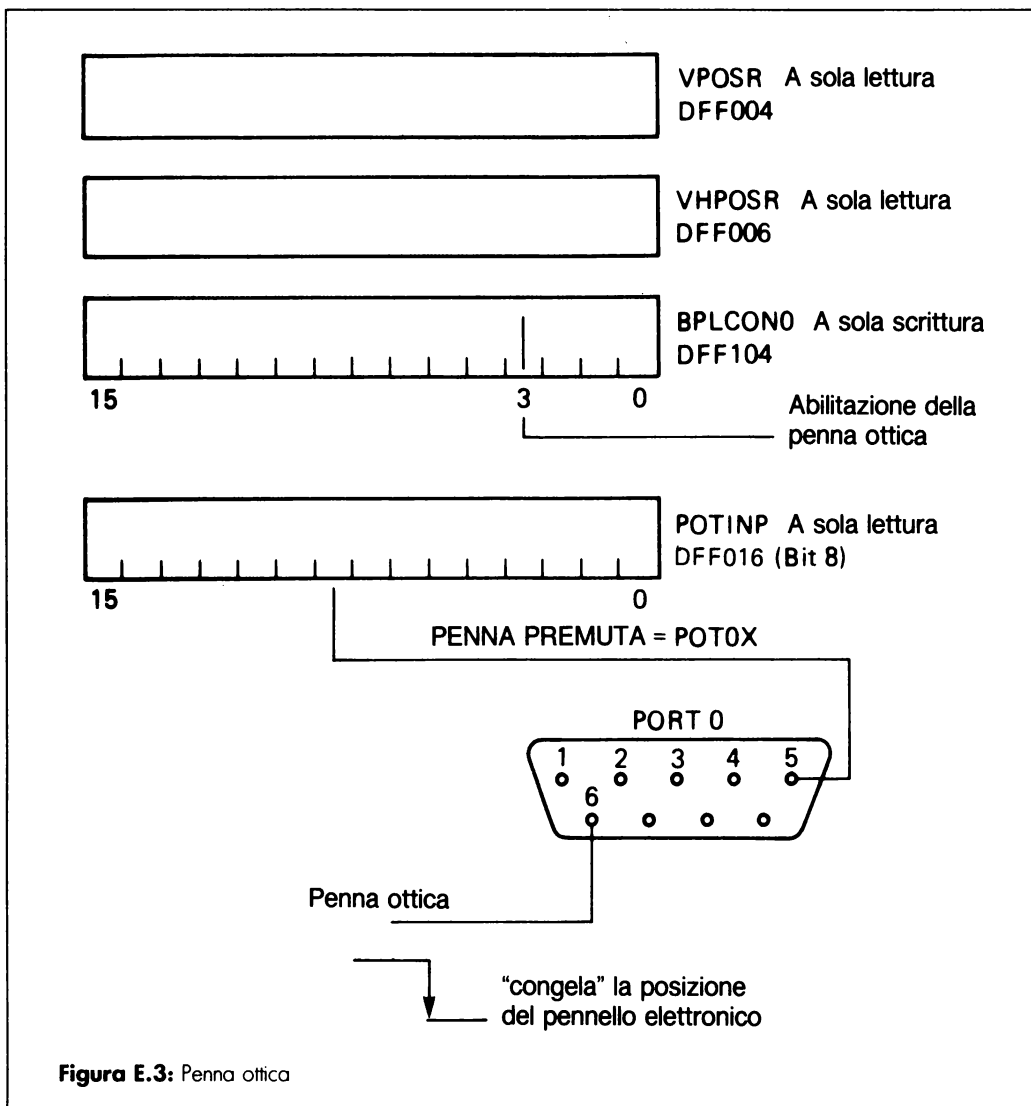
1	GND	18	DIRB
2	CHNG*	19	GND
3	GND	20	STEPB*
4	MTR0D*(led)	21	GND
5	GND	22	DKWDB*
6	NC	23	GND
7	GND	24	DKWEB*
8	INDEX*	25	GND
9	GND	26	TK0*
10	SEL0B*	27	GND
11	GND	28	WPRO*
12	NC	29	GND
13	GND	30	DKRD*
14	NC	31	GND
15	GND	32	SIDEB*
16	MTR0D*	33	GND
17	GND	34	RDY*

Alimentazione disk drive interno – 4 pin (J13)

1	+12V (alcuni disk drive utilizzano solo +5V)
2	GND
3	GND
4	+5V

Figura E.1: Contatori proporzionali

Figura E.2: Lettura dei pulsanti Fire

Figura E.3: Penna ottica

F COMPLEX INTERFACE ADAPTERS (CIA)

I chip 8520 (CIA)

Ogni Amiga contiene due chip 8520 detti CIA. Entrambi possiedono 16 pin generici di input/output, un registro di shift seriale, tre timer, un pin di solo output e uno di solo input. Molte funzioni dell'Amiga sono affidate a questi chip.

Ogni 8520 possiede 16 registri, accessibili sia in lettura che in scrittura. Le tavole seguenti elencano i registri disponibili e i loro indirizzi di accesso.

Mappa dei registri dei chip

RS3	RS2	RS1	RS0	NUM.	NOME	DESCRIZIONE
0	0	0	0	$0	pra	Registro dati A
0	0	0	1	$1	prb	Registro dati B
0	0	1	0	$2	ddra	Registro di direzione dati A
0	0	1	1	$3	ddrb	Registro di direzione dati B
0	1	0	0	$4	talo	Timer A, byte meno significativo
0	1	0	1	$5	tahi	Timer A, byte più significativo
0	1	1	0	$6	tblo	Timer B, byte meno significativo
0	1	1	1	$7	tbhi	Timer B, byte più significativo
1	0	0	0	$8	todlow	Bit 7-0
1	0	0	1	$9	todmid	Bit 15-8
1	0	1	0	$A	todhi	Bit 23-16
1	0	1	1	$B		Nessuna connessione
1	1	0	0	$C	sdr	Registro di shift seriale
1	1	0	1	$D	icr	Registro di controllo degli interrupt
1	1	1	0	$E	cra	Registro di controllo A
1	1	1	1	$F	crb	Registro di controllo B

Mappa degli indirizzi del chip CIAA

INDIR.	NOME	FUNZIONE
BFE001	pra	Bit 7-0 = /FIR1 /FIR0 /RDY /TK0 /WPRO /CHNG /LED OVL.
BFE101	prb	Porta parallela.
BFE201	ddra	Direzione dati per la porta A (BFE001). 1 = output (normalmente a $03).
BFE301	ddrb	Direzione dati per la porta B (BFE101). 1 = output.
BFE401	talo	Byte meno significativo del timer A (0,709379 PAL; 0,715909 NTSC).
BFE501	tahi	Byte più significativo del timer A.
BFE601	tblo	Byte meno significativo del timer B (0,709379 PAL; 0,715909 NTSC).
BFE701	tbhi	Byte più significativo del timer B.
BFE801	todlo	Bit 7-0 del timer a 50/60 Hz.
BFE901	todmid	Bit 15-8 del timer a 50/60 Hz.
BFEA01	todhi	Bit 23-16 del timer a 50/60 Hz.
BFEB01		Non utilizzato.
BFEC01	sdr	Registro di shift seriale (collegato alla tastiera).
BFED01	icr	Registro di controllo interrupt.
BFEE01	cra	Registro di controllo A.
BFEF01	crb	Registro di controllo B.

Nota: il chip CIAA può dare origine all'interrupt INT2.

Mappa degli indirizzi del chip CIAB

INDIR.	NOME	FUNZIONE
BFD000	pra	Bit 7-0 = /DTR /RTS /CD /CTS /DSR SEL POUT BUSY.
BFD100	prb	Bit 7-0 = /MTR /SEL3 /SEL2 /SEL1 /SEL0 /SIDE DIR /STEP.
BFD200	ddra	Direzione dati per la porta A (BFD000). 1 = output (normalmente a $FF).
BFD300	ddrb	Direzione dati per la porta B (BFD100). 1 = output (normalmente a $FF).
BFD400	talo	Byte meno significativo del timer A (0,709379 PAL; 0,715909 NTSC).
BFD500	tahi	Byte più significativo del timer A.
BFD600	tblo	Byte meno significativo del timer B (0,709379 PAL; 0,715909 NTSC).
BFD700	tbhi	Byte più significativo del timer B.
BFD800	todlo	Bit 7-0 del timer di sincronismo orizzontale.
BFD900	todmid	Bit 15-8 del timer di sincronismo orizzontale.
BFDA00	todhi	Bit 23-16 del timer di sincronismo orizzontale.
BFDB00		Non utilizzato.
BFDC00	sdr	Registro di shift seriale (non utilizzato).
BFDD00	icr	Registro di controllo interrupt.
BFDE00	cra	Registro di controllo A.
BFDF00	crb	Registro di controllo B.

Nota: il chip CIAB può dare origine all'interrupt INT6.

Nota: il sistema operativo utilizza normalmente per i propri compiti molti dei timer disponibili nei chip 8520:

CIAA, timer A   Utilizzato per l'interfacciamento con la tastiera.

CIAA, timer B	Utilizzato dall'Exec per lo scambio di controllo dei task, gli interrupt e altre temporizzazioni.
CIAA, TOD	Timer a 50/60 Hz utilizzato dal timer.device. L'A1000 utilizza la frequenza della corrente di alimentazione. L'A500 il sincronismo verticale. L'A2000 possiede un jumper per la selezione di una di queste due possibilità.
CIAB, timer A	Non utilizzato, disponibile per i programmi.
CIAB, timer B	Non utilizzato, disponibile per i programmi.
CIAB, TOD	Utilizzato dalla graphics.library per seguire la posizione del pennello elettronico. Questo timer è regolato sul sincronismo orizzontale, ed è usato per sincronizzare le operazioni grafiche con la posizione del pennello elettronico.

Nota: le precedenti edizioni di questa tabella non erano corrette.

Descrizione delle funzioni dei registri

PORTE di I/O (PRA, PRB, DDRA, DDRB)

Le porte A e B consistono entrambe di un registro dati a 8 bit (PR) e di un registro di direzione, sempre a 8 bit (DDR). Se un certo bit del registro DDR è a 1, la posizione corrispondente del registro PR diventa un output. Se, al contrario, il bit di DDR è a 0, il relativo bit di PR risulta definito come input.

Nella lettura del registro PR, il valore letto corrisponde sempre all'effettivo stato dei pin di I/O (PA0-PA7, PB0-PB7), comunque siano definiti.

Entrambe le porte sono progettate per garantire la compatibilità con gli standard CMOS e TTL.

In aggiunta alle normali funzioni di I/O, i pin PB6 e PB7 possono anche svolgere funzioni di output per il timer.

PROTOCOLLO D'INTERFACCIAMENTO

Il protocollo usato nel trasferimento di dati si basa sull'uso del pin di output PC e del pin di input FLAG. Il segnale PC diventa basso nel terzo ciclo successivo all'accesso alla porta B. Questo segnale può essere usato per indicare "dati pronti sulla porta B" o "dati accettati dalla porta B". Per trasferire 16 bit di dati tramite questo protocollo, utilizzando entrambe le porte A e B, si deve sempre leggere o scrivere per prima la porta A. FLAG è un input negativo che può essere utilizzato per ricevere l'output PC da un altro 8520, oppure come input generico. Qualunque transizione negativa della linea collegata a FLAG provoca l'impostazione del bit di interrupt FLAG.

REG.	NOME	D7	D6	D5	D4	D3	D2	D1	D0
0	PRA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
1	PRB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
2	DDRA	DPA7	DPA6	DPA5	DPA4	DPA3	DPA2	DPA1	DPA0
3	DDRB	DPB7	DPB6	DPB5	DPB4	DPB3	DPB2	DPB1	DPB0

TIMER (TIMER A, TIMER B)

Ogni timer consiste di un registro contatore a 16 bit a sola lettura e di un valore di latch a sola scrittura, sempre a 16 bit. Accedere in scrittura a un timer provoca pertanto l'impostazione del suo valore di latch, mentre la lettura fornisce il contenuto del registro contatore.

Il valore di latch viene anche detto "prescalare", in quanto viene usato come valore iniziale per il conto alla rovescia che termina quando si raggiunge il valore 0. Questo prescalare è un divisore della frequenza di clock del sistema. I timer possono essere usati indipendentemente l'uno dall'altro, oppure collegati insieme per operazioni più complesse. Vari modi operativi consentono di generare intervalli di tempo più lunghi, impulsi di lunghezza variabile, treni di impulsi e forme d'onda a frequenza variabile. Utilizzando il segnale di input CNT, i timer possono contare gli impulsi provenienti da una sorgente esterna o misurarne la frequenza, la durata e l'intervallo di ritardo.

A ogni timer è associato un registro i cui bit permettono il controllo individuale di ciascuna delle seguenti funzioni:

Avvio/Arresto

Un bit di controllo consente al 68000 di avviare o arrestare il timer in qualunque momento.

PB on/off

Un altro bit di controllo consente all'ouput del timer di apparire sulla porta B (sul pin 6 per il timer A e sul pin 7 per il timer B) del chip CIA corrispondente. Qualunque sia la condizione in uso, impostata tramite il registro DDRB, i relativi pin PB vengono forzati a diventare degli output.

Alternato/Impulso

Un bit di controllo permette di decidere quale output dev'essere inviato alla porta B quando il pin PB on/off è attivato (on). Tramite questo bit è possibile decidere se al termine di ogni conteggio viene inviato un singolo impulso positivo della durata di un ciclo oppure se lo stato del segnale dev'essere invertito fino al termine del conteggio successivo.

Singolo/Continuo

Un altro bit di controllo seleziona uno dei due seguenti modi operativi. Nel "modo singolo", il timer decrementa il contatore fino a raggiungere il valore zero, genera un interrupt, ricarica il valore del contatore con il valore di latch e quindi si ferma. Nel "modo continuo", invece, quando il timer raggiunge il valore zero, genera un interrupt, ricarica nel contatore il valore di latch e quindi riparte ripetendo la procedura all'infinito.

In modo singolo, un accesso in scrittura al byte più significativo del timer (il registro 5 per il timer A, il registro 7 per il timer B) provoca il trasferimento del valore di latch nel contatore e dà il via al conteggio, qualunque sia lo stato del bit di avvio/arresto.

Caricamento forzato

Un bit di strobe permette di forzare il valore di latch nel contatore in qualunque istante, che il timer sia attivo o no.

MODALITÀ DI INPUT

I bit di controllo permettono inoltre di selezionare il clock da usare per decrementare il contatore. Il timer A può utilizzare gli impulsi del clock 02 o gli impulsi esterni applicati al pin CNT. Il timer B può utilizzare gli impulsi del clock 02, gli impulsi CNT esterni, tutti gli impulsi di conteggio terminato provenienti dal timer A, oppure gli impulsi provenienti dal timer A ma solo quando il segnale CNT è alto.

Il valore di latch viene caricato nel contatore al termine di ogni conteggio, oppure tramite il bit di caricamento forzato, o ancora in seguito all'impostazione del byte alto del valore di latch a timer fermo. Se il timer non è fermo, invece, la scrittura di questo byte provoca l'impostazione del valore di latch ma non il suo caricamento nel contatore.

NOMI DEI BIT DEI REGISTRI IN LETTURA

REG.	NOME	D7	D6	D5	D4	D3	D2	D1	D0
4	TALO	TAL7	TAL6	TAL5	TAL4	TAL3	TAL2	TAL1	TAL0
5	TAHI	TAH7	TAH6	TAH5	TAH4	TAH3	TAH2	TAH1	TAH0
6	TBLO	TBL7	TBL6	TBL5	TBL4	TBL3	TBL2	TBL1	TBL0
7	TBHI	TBH7	TBH6	TBH5	TBH4	TBH3	TBH2	TBH1	TBH0

NOMI DEI BIT DEI REGISTRI IN SCRITTURA

REG.	NOME	D7	D6	D5	D4	D3	D2	D1	D0
4	TALO	PAL7	PAL6	PAL5	PAL4	PAL3	PAL2	PAL1	PAL0
5	TAHI	PAH7	PAH6	PAH5	PAH4	PAH3	PAH2	PAH1	PAH0
6	TBLO	PBL7	PBL6	PBL5	PBL4	PBL3	PBL2	PBL1	PBL0
7	TBHI	PBH7	PBH6	PBH5	PBH4	PBH3	PBH2	PBH1	PBH0

Orologio "Time of Day" (TOD)

Il timer TOD è un contatore a 24 bit. Tutte le transizioni positive sul pin corrispondente causano il suo incremento. Al pin è associata una resistenza di pull-up passiva.

Esiste inoltre un "allarme" programmabile che può dare origine a un interrupt al momento desiderato. Gli indirizzi corrispondenti all'allarme sono gli stessi del timer. Il modo di accesso è controllato da un bit del registro di controllo. L'accesso all'allarme avviene soltanto in scrittura: la lettura di un registro TOD fornisce il valore del timer in quell'istante, qualunque sia lo stato del bit di accesso all'allarme.

È necessario seguire una particolare procedura per impostare e leggere il timer TOD. Esso viene automaticamente bloccato da un accesso in scrittura a un suo qualsiasi registro, e riparte

solo in seguito a un successivo accesso in scrittura al registro contenente il byte di dati meno significativo. Ciò permette di assicurarsi che il timer TOD parta sempre esattamente dal valore desiderato.

Dal momento che durante le operazioni di lettura dei registri potrebbe verificarsi un riporto da un byte a quello immediatamente più significativo, il timer TOD possiede una funzione di latch che per tutta la durata del processo di lettura mantiene costanti le informazioni presenti nei suoi registri. Questa condizione di "congelamento" si verifica nel caso di un accesso in lettura al byte più significativo, e perdura fino alla lettura del byte meno significativo. Mentre si trova in questa condizione, il timer continua a funzionare: è soltanto il valore in lettura che viene bloccato. Se si desidera leggere uno solo di questi registri, ovviamente non vi è nessun problema di riporto, e il registro può essere letto "al volo", pur tenendo sempre presente che ogni lettura del byte più significativo dev'essere seguita da quella del byte meno significativo per disabilitare il "congelamento" descritto.

NOMI DEI BIT PER L'ACCESSO AL TIMER O ALL'ALLARME

REG.	NOME								
8	TODLO	E7	E6	E5	E4	E3	E2	E1	E0
9	TODMID	E15	E14	E13	E12	E11	E10	E9	E8
A	TODHI	E23	E22	E21	E20	E19	E18	E17	E16

SCRITTURA	
CRB7 = 0	TOD
CRB7 = 1	ALLARME

Registro di shift seriale (SDR)

Questo registro è una porta seriale costituita da un registro di shift a 8 bit sincrono. Un bit di controllo seleziona il modo input o output. Nell'Amiga uno di questi registri viene utilizzato per la lettura dei dati provenienti dalla tastiera, mentre l'altro è a disposizione dei programmi applicativi. La porta seriale compatibile RS232 è invece contenuta nel chip Paula. Si consulti il Capitolo 8 per maggiori informazioni al riguardo.

MODO INPUT

In modo input, i dati ricevuti sul pin SP vengono inseriti nel registro di shift a ogni transizione positiva del pin CNT. Dopo otto transizioni, i dati contenuti in questo registro vengono trasferiti nel registro dati seriale e viene generato un interrupt.

MODO OUTPUT

In modo output, il timer A viene usato come regolatore della velocità di trasmissione. I dati vengono inviati sul pin SP a metà della velocità di underflow del timer A. La massima velocità teoricamente raggiungibile è quindi quella del clock 02 divisa per 4, ma in pratica si è limitati dalle capacità delle linee di trasferimento dati e dalla velocità con cui il dispositivo ricevente

può rispondere ai dati inviati.

Per iniziare la trasmissione occorre innanzitutto predisporre il timer A in modo continuo, e quindi avviarlo. La trasmissione dei dati ha luogo in seguito alla loro scrittura nel registro dati seriale. Il segnale di clock ricavato dal timer A appare come output sul pin CNT. I dati contenuti nel registro seriale vengono trasferiti nel registro di shift e di qui, a ogni transizione del pin CNT, viene inviato in output un bit di dati sul pin SP. Questi dati sono validi a partire dalla successiva transizione negativa del pin CNT e lo rimangono fino a quella ancora seguente.

Dopo otto impulsi CNT, viene generato un interrupt per indicare che possono essere inviati altri dati. Se prima di questo interrupt nel registro dati seriale era stato inserito un nuovo valore, i nuovi dati vengono automaticamente trasferiti nel registro di shift e la trasmissione continua.

Se dopo l'ottavo impulso di CNT non vengono inviati altri dati, CNT ritorna alto, e il pin SP rimane al valore corrispondente all'ultimo dato trasmesso.

I dati vengono trasmessi a partire dal bit più significativo e così dovrebbero anche apparire nel modo input.

CARATTERISTICHE BIDIREZIONALI

Le capacità bidirezionali del registro di shift e del clock CNT permettono di collegare parecchi 8520 a un solo bus seriale di comunicazione, con un particolare 8520 che agisce come unità master, sorgente dei dati e del clock, e gli altri che agiscono come unità asservite. L'hardware relativo ai pin CNT e SP è appositamente studiato per permettere questo tipo di collegamento. Il protocollo relativo alla selezione dell'unità master può essere trasmesso tramite questo stesso bus seriale oppure tramite linee dedicate.

REG.	NOME	D7	D6	D5	D4	D3	D2	D1	D0
C	SDR	S7	S6	S5	S4	S3	S2	S1	S0

Registro di controllo degli interrupt (ICR)

Vi sono cinque sorgenti di interrupt nei chip 8520:

- Underflow del timer A (il conteggio raggiunge lo zero).
- Underflow del timer B.
- Allarme del timer TOD.
- Porta seriale piena o vuota.
- Flag.

Un registro fornisce la possibilità di mascherare queste sorgenti e di avere informazioni sugli interrupt. Consiste infatti di una maschera a sola scrittura e di un registro dati a sola lettura, entrambi allo stesso indirizzo. Un interrupt risulta abilitato se il corrispondente bit della maschera è impostato a 1. La richiesta di un interrupt fa sì che il corrispondente bit del registro dati venga impostato a 1. Se poi l'interrupt richiesto è anche abilitato, ciò provoca l'impostazione del bit più significativo (IR) del registro dati e provoca la transizione negativa del segnale IRQ. In un sistema multichip basta esaminare il bit IR per sapere quale chip ha generato un certo interrupt.

Il processo di lettura del registro dati ne provoca anche l'azzeramento, e la linea IRQ ritorna

allo stato alto. A causa di questo fatto, occorre assicurarsi che le proprie routine di gestione degli interrupt possano mantenere tutti i bit che erano impostati nel registro dati al momento della lettura e reagire opportunamente allo loro impostazione. Con un'adeguata programmazione, è possibile mescolare così la gestione "manuale" e quella "automatica" di questi interrupt.

È possibile impostare o azzerare uno o più bit della maschera senza interferire con gli altri. Per indicare, tramite il valore 1, quali bit devono essere modificati si utilizzano i bit 6-0 del valore in scrittura, mentre il bit 7 serve per specificare come devono essere modificati. Se il bit 7 è a 1, i bit vengono impostati, in caso contrario vengono azzerati.

Per abilitare la generazione di un determinato interrupt, occorre che il corrispondente bit della maschera venga impostato a 1.

Per esempio, supponiamo di voler abilitare l'interrupt relativo al timer A del chip CIAA, disabilitando contemporaneamente tutti gli altri. Ecco una possibile sequenza di istruzioni:

```
INCLUDE "hardware/cia.i"
XREF    _ciaa                   ;Da amiga.lib
lea     _ciaa,a0                ;Definito in amiga.lib
move.b  #%01111110,ciaicr(a0) ;Azzera i bit 6-1
move.b  #%10000001,ciaicr(a0) ;Imposta il bit 0
```

REGISTRO DI CONTROLLO DEGLI INTERRUPT IN LETTURA

REG.	NOME	D7	D6	D5	D4	D3	D2	D1	D0
D	ICR	IR	0	0	FLG	SP	ALRM	TB	TA

REGISTRO DI CONTROLLO DEGLI INTERRUPT IN SCRITTURA

REG.	NOME	D7	D6	D5	D4	D3	D2	D1	D0
D	ICR	S/C	x	x	FLG	SP	ALRM	TB	TA

Registri di controllo

Ci sono due registri di controllo in ogni 8520: CRA e CRB. CRA è associato al timer A e CRB al timer B. Il loro formato è illustrato nel paragrafo seguente.

REGISTRO DI CONTROLLO A

BIT	NOME	FUNZIONE
0	START	1 = avvio del timer A, 0 = arresto del timer A. Questo bit viene automaticamente azzerato al termine del conteggio in modo singolo.
1	PBON	1 = output sul pin PB6, 0 = PB6 normale.
2	OUTMODE	1 = alternato, 0 = impulso.
3	RUNMODE	1 = singolo, 0 = continuo.

4	LOAD	1 = strobe, provoca il caricamento del valore di latch nel contatore. In lettura restituisce sempre 0. La scrittura di uno 0 non ha alcun effetto.
5	INMODE	1 = conteggio delle transizioni positive di CNT, 0 = impulsi ricavati dal clock 02.
6	SPMODE	1 = porta seriale in modo output (clock = CNT), 0 = porta seriale in modo input (richiede un clock esterno).
7	Non utilizzato	

MAPPA DEI BIT DEL REGISTRO CRA

REG.	NOME	x	SPMODE	INMODE	LOAD	RUNMODE	OUTMODE	PBON	START
E	CRA	x	0 = input	0 = 02	1 = strobe	0 = cont.	0 = altern.	0 = PB6OFF	0 = stop
		x	1 = output	1 = CNT		1 = singolo	1 = impulso	1 = PB6ON	1 = start

Tutti i bit non utilizzati non vengono modificati in scrittura e vengono forzati a 0 in lettura.

REGISTRO DI CONTROLLO B

BIT	NOME	FUNZIONE
0	START	1 = avvio del timer B, 0 = arresto del timer B. Questo bit viene automaticamente azzerato al termine del conteggio in modo singolo.
1	PBON	1 = output sul pin PB7, 0 = PB7 normale.
2	OUTMODE	1 = alternato, 0 = impulso.
3	RUNMODE	1 = singolo, 0 = continuo.
4	LOAD	1 = strobe, provoca il caricamento del valore di latch nel contatore. In lettura resituisce sempre 0. La scrittura di uno 0 non ha alcun effetto.
6,5	INMODE	I bit CRB6 e CRB5 selezionano uno dei quattro possibili modi di input:

CRB6	CRB5	Modo selezionato
0	0	Impulsi del clock 02
0	1	Transizioni positive del pin CNT
1	0	Impulsi di underflow del timer A
1	1	Impulsi di underflow del timer A quando il segnale CNT è alto

7	ALARM	1 = l'accesso in scrittura al timer TOD imposta l'allarme, 0 = l'accesso in scrittura al timer TOD imposta il timer stesso. L'accesso in lettura fornisce comunque il contenuto del timer TOD.

MAPPA DEI BIT DEL REGISTRO CRB

REG.	NOME	ALARM	INMODE	LOAD	RUNMODE	OUTMODE	PBON	START
F	CRB	0 = TOD	00 = 02	1 = strobe	0 = cont.	0 = PB7OFF	0 = altern.	0 = stop
		1 = alarm	01 = CNT		1 = singolo	1 = PB7ON	1 = impulso	1 = start
			10 = timer A					
			11 = CNT + timer A					

Tutti i bit non utilizzati non vengono modificati in scrittura e vengono forzati a 0 in lettura.

Assegnazione dei segnali alle porte

Questa parte descrive quali segnali sono collegati alle varie porte dei chip 8520.

Indirizzo BFEr01 (A12*) (int2)

PA7	FIRE1*	Porta 1, pin 6.
PA6	FIRE0*	Porta 0, pin 6.
PA5	RDY*	Drive pronto.
PA4	TK0*	Traccia 00.
PA3	WPRO*	Disco protetto da scrittura.
PA2	CHNG*	Disco estratto dal drive.
PA1	LED*	Led di alimentazione/filtro audio (1 = disinserito).
PA0	OVL	Bit di overlay di ROM e RAM.
SP	KDAT	Dati dalla tastiera.
CNT	KCLK	Clock della tastiera.
PB7-PB0	DATA7-0	Dati dell'interfaccia parallela.
PC	DRDY*	Controllo dell'interfaccia parallela.
F	ACK*	Controllo dell'interfaccia parallela.

Indirizzo BFDr00 (A13*) (int6)

PA7	DTR*	
PA6	RTS*	
PA5	CD*	
PA4	CTS*	
PA3	DSR*	
PA2	SEL	Controllo dell'interfaccia parallela.
PA1	POUT	Fine carta.
PA0	BUSY	Busy
SP	BUSY	Commodore.
CNT	POUT	Commodore.
PB7	MTR*	Motore del disk drive.
PB6	SEL3*	Selezione disk drive 3.
PB5	SEL2*	Selezione disk drive 2.
PB4	SEL1*	Selezione disk drive 1.
PB3	SEL0*	Selezione disk drive 0.
PB2	SIDE*	Selezione della faccia.

PB1	DIR	Direzione.
PB0	STEP*	Step (minimo 3 millisecondi).
PC	Non utilizzato	
F	INDEX*	

ESEMPIO

```
;Un esempio completo di temporizzazione tramite 8520. La luce
;del led di alimentazione viene fatta lampeggiare a intervalli
;di esattamente 3 millisecondi. Attenzione! Il programma
;prende il completo controllo della macchina.
;
;Le frequenze dei cristalli dell'Amiga sono:
;       PAL   28,37516 MHz
;       NTSC  28,63636 MHz
;
;I due timer a 16 bit vengono decrementati a 1/10 del clock
;della CPU o 0,709379 MHz, che corrispondono a 1,4096837
;microsecondi per decremento. In NTSC il conteggio e'
;leggermente piu' veloce: 0,715909 MHz.
;
;Per attendere 1/100 di secondo (ovvero 10 mila microsecondi)
;il registro va pertanto impostato a:
;10000/1,4096837 = 7094
;
;Per attendere 3 millisecondi (ovvero 3 mila microsecondi)
;il registro va invece impostato a:
;3000/1,4096837 = 2128
;
      INCLUDE "hardware/cia.i"
      INCLUDE "hardware/custom.i"

      xref    _ciaa
      xref    _ciab
      xref    _custom

      lea     _custom,a3                ;Base dei chip custom
      lea     _ciaa,a4                  ;Indirizzo del chip CIAA
      lea     _ciab,a5                  ;Indirizzo del chip CIAB

      move.w  #$7FFF,intena(a3)         ;Disabilita gli interrupt
;
;Impostazione dei bit per l'uso del timer A in modo singolo.
;
      move.b  ciacra(a4),d0             ;Registro di controllo A
      and.b   #%11000000,d0
      or.b    #%00001000,d0
      move.b  d0,ciacra(a4)
      move.b  #%01111111,ciaicr(a4)     ;Disabilita gli interrupt
                                        ;del chip 8520
```

```
;
;Imposta il valore iniziale (prima il byte meno significativo)
;
TIME equ     2128
     move.b  #(TIME&$FF),ciatalo(a4)
     move.b  #(TIME>>8),ciatahi(a4)
;
;Attende il termine del conteggio
;
attesa:
     btst.b  #0,ciaicr(a4)
     beq.s   attesa
     bchg.b  #CIAB_LED,ciapra(a5)
     bset.b  #0,ciacra(a4)           ;Riavvia il timer
     bra.s   Attesa

     END
```

Dettagli sui collegamenti hardware

L'hardware del sistema seleziona i chip CIA ogniqualvolta i tre bit più significativi dell'indirizzo sono 101. Inoltre CIAA viene selezionato quando A12 è a 0 e A13 è a 1. CIAB quando A12 è a 1 e A13 è a 0. CIAA comunica sui bit di dati 7-0, CIAB sui bit 15-8.

A11, A10, A9 e A8 sono usati per selezionare uno dei 16 registri interni, indicati da "r" nell'indirizzo. Tutti gli altri bit non hanno importanza. Pertanto, il chip CIAA viene selezionato tramite gli indirizzi 101x xxxx xx01 rrrr xxxx xxx0. Il CIAB, invece tramite 101x xxxx xx10 rrrr xxxx xxx1.

Tenendo presente la possibilità di futuri ampliamenti del sistema, sono stati scelti i seguenti indirizzi: CIAA = BFEr01; CIAB = BFDr00. Con questi indirizzi, i programmi devono usare istruzioni in formato byte e nessun altro tipo di accesso.

SEGNALI DI INTERFACCIAMENTO

02 – clock

Il clock 02 è un input TTL compatibile che viene usato nelle operazioni interne dei dispositivi dell'Amiga e per le comunicazioni con il bus dati del sistema. È collegato al clock "E" del 68000, il che corrisponde a 1/10 del clock della CPU, ovvero 0,709379 MHz in standard PAL, oppure 0,715909 MHz in NTSC.

CS – input di selezione del chip

L'input CS controlla l'attività del chip 8520. Un livello basso di CS mentre 02 è alto fa sì che il dispositivo risponda ai segnali R/W e alle linee di indirizzo (RS). Quando CS è alto, questi segnali non influiscono sull'8520. Il segnale CS viene normalmente attivato (basso) da un'appropriata combinazione delle linee`d'indirizzo.

R/W – input di lettura/scrittura

Il segnale R/W viene fornito in genere dalla CPU e controlla la direzione dei trasferimenti di dati con i chip 8520. Un segnale alto indica un'operazione di lettura (i dati provengono dal chip), un segnale basso il contrario.

RS3-RS0 – linee d'indirizzo

Queste linee selezionano uno dei 16 registri interni (si veda la mappa dei registri).

DB7-DB0 – bus dati di I/O

Questi otto pin sono dedicati al trasferimento dei dati tra i chip 8520 e il bus dati del sistema: sono input ad alta impedenza, a meno che il segnale CS non sia basso e R/W e 02 siano alti, il che permette la lettura. Durante la lettura i buffer di output del bus dati vengono abilitati e i dati provenienti dal registro selezionato vengono trasferiti sul bus dati di sistema.

IRQ – output di richiesta di interrupt

IRQ è un pin di output collegato alle linee di interrupt della CPU. Una resistenza esterna di pull-up mantiene il segnale alto, permettendo così la connessione di più segnali IRQ. IRQ è normalmente disattivato (alta impedenza), e viene attivato basso come si è già visto nella descrizione del registro di interrupt.

RES – reset input

Un segnale basso sul pin RES provoca il reset di tutti i registri interni. I pin delle porte PRA e PRB divengono di input, e tutti i registri corrispondenti vengono azzerati (una lettura fornisce però valori tutti alti a causa delle resistenze passive di pull-up). I registri di controllo dei timer vengono azzerati, e i contatori vengono inizializzati tutti a 1. Gli altri registri vengono azzerati.

Il protocollo AUTOCONFIG è stato progettato per permettere l'assegnazione dinamica degli spazi d'indirizzamento disponibili alle schede di espansione, eliminando la necessità d'interventi esterni tramite jumper o altro. Durante il reset, le schede appaiono una per volta all'indirizzo $E80000, con le informazioni d'identificazione (quasi tutte in complemento a 1) contenute nei nibble più significativi delle prime $40 word ($80 byte) della scheda stessa. Queste informazioni comprendono le dimensioni della scheda, eventuali preferenze per il suo spazio d'indirizzamento, il tipo della scheda (espansione di memoria o altro), e un numero d'identificazione assegnato dalla Commodore tramite il Commodore Amiga Technical Support.

Ogni scheda deve contenere nell'hardware di configurazione un indirizzo di latch suddiviso nei nibble agli offset $0048 e $004A. Quando in questo registro vengono inseriti i campi da A16 ad A23 dell'indirizzo assegnato dal sistema come indirizzo base della scheda, quest'ultima appare al nuovo indirizzo e deve quindi trasmettere un segnale, chiamato CONFIG-OUT, in modo da lasciare il posto (ovvero l'indirizzo $E80000) alla scheda successiva perché venga configurata a sua volta. Per rendere più economica una scheda di espansione, i suoi registri di scrittura possono essere organizzati come un unico registro da 1 byte o come due registri da 1 nibble. In tal caso, il nibble meno significativo ($4A) dev'essere bloccato fino a quando non è stato impostato il nibble più significativo ($48). Ciò permette all'algoritmo seguente di funzionare con entrambi i tipi di scheda:

> Scrivere il nibble meno significativo all'offset $4A.
> Scrivere l'intero byte dell'indirizzo all'offset $48.

Inoltre, molte schede possono essere "zittite" (shut-up): trasmettono cioè il segnale CONFIG-OUT e cessano di rispondere, accedendo in scrittura all'offset $004C della scheda. Un bit nel nibble all'offset $0008 indica se la scheda permette o meno lo shut-up.

Tutte le schede di espansione per l'Amiga dovrebbero seguire il protocollo AUTOCONFIG. Informazioni più approfondite sul disegno e la costruzione di schede AUTOCONFIG possono essere richieste al Commodore Amiga Technical Support.

Il sistema operativo dell'Amiga contiene tutte le funzioni necessarie per la configurazione di

schede tramite driver residenti su disco. A partire dalla versione 1.3, il sistema operativo permette anche l'inizializzazione tramite driver in ROM. Come regola generale, i programmi non dovrebbero tentare di configurare personalmente le schede di espansione, ma dovrebbero permettere al sistema di farlo in modo automatico. Molte schede contengono registri in grado di causare gravi danni; per esempio, i dati presenti su un hard disk potrebbero essere persi se la scheda non viene configurata correttamente.

Tuttavia, certi programmi che prendono il completo controllo della macchina potrebbero avere la necessità di configurare una scheda come espansione di memoria o hardware dedicato: questi programmi dovrebbero limitarsi a configurare le schede di espansione RAM (schede che chiedono di essere aggiunte alla lista della memoria libera) o quelle per cui sono stati appositamente scritti. Tutte le altre schede dovrebbero essere "zittite" se permettono lo shut-up oppure configurate e quindi ignorate se non lo permettono (vi sono molte schede che non prevedono lo shut-up).

Il processo di configurazione dovrebbe essere tentato soltanto dai programmi che al reset prendono il controllo completo del sistema. La presenza di una scheda che attende di essere configurata può essere determinata leggendo i nibble all'inizio dello spazio di memoria AUTOCONFIG e confrontandoli con quelli previsti dalle schede.

L'architettura AUTOCONFIG richiede che le schede da configurare appaiano a indirizzi multipli delle loro dimensioni. Per esempio, una scheda di espansione da 1 MB dev'essere configurata a un indirizzo che sia un multiplo intero di 1 MB. Vi sono però due eccezioni a questa regola: schede di espansione da 4 MB possono anche apparire agli indirizzi \$200000 e \$600000 oltre che ai multipli di 4 MB, e schede da 8 MB possono apparire all'indirizzo \$200000. Queste eccezioni sono necessarie poiché gli 8 MB riservati come spazio di espansione negli Amiga attuali iniziano dall'indirizzo \$200000.

Debug delle schede AUTOCONFIG

Se c'è un errore nelle informazioni di configurazione, una scheda potrebbe essere ignorata, potrebbe essere "zittita", ma potrebbe anche andare in crash in modo tale da rendere difficile scoprirne il motivo. Vi è un meccanismo abbastanza semplice che permette di controllare l'esattezza di queste informazioni: è sufficiente collegare il segnale CONFIGIN* a uno switch. Quando questo switch chiude il circuito, il segnale CONFIGIN* può passare dal bus alla scheda che pertanto risponde al processo AUTOCONFIG. Se lo switch invece è aperto il processo di configurazione per la scheda viene disabilitato.

Avviando il sistema con lo switch aperto, la scheda risulta invisibile per il software sistema. Dopo aver mandato in esecuzione un debugger si chiude lo switch, e la scheda appare all'indirizzo \$E80000 nelle stesse condizioni in cui viene vista dal sistema nel processo di configurazione. È possibile quindi confrontare i valori effettivi con quelli previsti.

La scheda di cui effettuare il debug dev'essere l'ultima presente nel sistema (la più vicina agli slot PC, ovvero la più lontana dal gruppo di alimentazione). Schede "a valle" di questa non vengono configurate dal sistema.

Tavola di specifica degli indirizzi

La tavola seguente descrive le informazioni d'identificazione e i registri AUTOCONFIG che appaiono nei primi \$80 byte di una scheda durante la configurazione.

- Le informazioni d'identificazione si trovano nei due nibble più significativi degli indirizzi pari all'inizio della scheda. Per esempiò, le prime due word potrebbero contenere il valore $Cxxx 1xxx. L'informazione valida è data, in questo caso, dai valori $C (nibble più significativo della word all'offset $00) e $1 (nibble più significativo della word all'offset $02). Gran parte delle informazioni vengono ottenute tramite combinazioni di diversi nibble, con i nibble più significativi all'indirizzo più basso.

- Tutti i nibble d'informazione, a eccezione di quelli agli indirizzi $00/02 e $40/42 sono memorizzati in complemento a 1 e devono subire un'operazione di OR esclusivo con il valore $F prima di essere interpretati secondo la tavola seguente. I nibble non utilizzati (i tre meno significativi di ogni word) possono contenere qualunque valore. Tutti i valori in scrittura, invece, incluso l'indirizzo assegnato, vengono scritti in forma normale.

- Tutti gli indirizzi che compaiono nella Tavola G-1 sono offset dall'indirizzo di base $E80000, cosicché l'indirizzo $02 corrisponde a $E80002, $04 a $E80004, e così via.

La numerazione dei bit (7 6 5 4 3 2 1 0) è da usarsi quando due nibble vengono interpretati come un unico byte. Fisicamente, ogni nibble è il nibble più significativo della word a quell'indirizzo (i bit 15, 14, 13, 12).

Tavola G-1: Tavola di specifica degli indirizzi

($08/0A)
Lettura in
complemento a 1
7 6 5 4
3 2 1 0
Riservato, dev'essere 0
0 = Questa scheda prevede lo shut-up
1 = Questa scheda ignora lo shut-up
0 = Qualunque spazio va bene
1 = È preferito uno spazio negli 8 MB di espansione
($0C/0E)
Lettura in
complemento a 1
7 6 5 4
3 2 1 0
Riservati. Devono essere 0
($10/12)
Lettura in
complemento a 1
7 6 5 4
3 2 1 0
Byte più significativo del numero
assegnato al produttore
dalla Commodore.
nibble più
significativo
nibble meno
significativo
($14/16)
Lettura in
complemento a 1
7 6 5 4
3 2 1 0
Byte meno significativo del numero
assegnato al produttore
dalla Commodore.
nibble più
significativo
nibble meno
significativo
($18/1A) 7 6 5 4 3 2 1 0 Numero di serie opzionale, primo byte
(più significativo)
($1C/1E) 7 6 5 4 3 2 1 0 Numero di serie opzionale, secondo byte
($20/22) 7 6 5 4 3 2 1 0 Numero di serie opzionale, terzo byte
($24/26) 7 6 5 4 3 2 1 0 Numero di serie opzionale, quarto byte
(meno significativo)
Lettura in
complemento a 1
($28/2A) 7 6 5 4 3 2 1 0 Byte più significativo del vettore ROM
opzionale
Lettura in
complemento a 1

($2C/2E)	7 6 5 4	3 2 1 0	Byte meno significativo del vettore ROM opzionale.
Lettura in complemento a 1			

(Se il bit "vettore ROM valido" contenuto nel nibble all'offet $00 è impostato, allora questo vettore rappresenta l'offset dall'indirizzo di base della scheda alle strutture del driver nella ROM della scheda).

($30/32)	7 6 5 4	3 2 1 0	Lettura: riservato, dev'essere 0
Lettura/scrittura in complemento a 1			Scrittura: reset opzionale dell'indirizzo di base della scheda all'indirizzo di pre-configurazione.

($34/36)	7 6 5 4	3 2 1 0	Riservato, dev'essere 0
($38/3A)	7 6 5 4	3 2 1 0	Riservato, dev'essere 0
($3C/3E)	7 6 5 4	3 2 1 0	Riservato, dev'essere 0
In complemento a 1			

Nota: l'uso dei registri $40/42 è una caratteristica opzionale che può essere utilizzata dalle schede che generano interrupt. Rende infatti possibile (grazie ad appositi server creati per la scheda) determinare se l'interrupt è stato generato proprio da quella scheda o da qualche altro componente hardware che utilizzi la stessa linea di interrupt.

($44/46)	7 6 5 4	3 2 1 0	Riservato, dev'essere 0.
Lettura/scrittura in complemento a 1			In scrittura non viene utilizzato.

($48/4A) Scrittura normale	7 6 5 4 — nibble più significativo	3 2 1 0 — nibble meno significativo	Indirizzo base assegnato. Questi bit vengono confrontati con le linee d'indirizzamento A23-A16 per determinare l'indirizzo base della scheda.
($4C/4E) Sola scrittura	7 6 5 4	3 2 1 0	Registro opzionale di shut-up. Una qualunque scrittura all'offset $4C fa sì che la scheda generi il segnale di CONFIG-OUT e non risponda più a nessun indirizzo fino al successivo RESET. Un bit nel nibble $08 indica se la scheda prevede o meno lo shut-up.
(da $50 a $7E) In complemento a 1			Riservato, dev'essere 0.

Si tenga presente che tutti i nibble eccetto $00/02 e $40/42 appaiono invertiti rispetto ai valori presentati nella tavola precedente. Un nibble che "dev'essere 0" verrà letto come $F, e saranno pure invertiti i vari flag e valori esadecimali (per esempio, un valore $1 apparirà come $E).

```
/* Esamina tutte le schede AUTOCONFIG del sistema */

#include "exec/types.h"
#include "libraries/configvars.h"

struct Library   *OpenLibrary();
struct ConfigDev *FindConfigDev();
struct Library   *ExpansionBase;

void main()
{
struct ConfigDev *cd=0;

ExpansionBase=OpenLibrary("expansion.library",0);
while(cd=FindConfigDev(cd,-1,-1)) /* Cerca qualunque ConfigDev */
    {
    printf("\n Struttura ConfigDev trovata all'indirizzo $%\n",cd);

    /* Questi valori vengono letti direttamente dalla scheda */
    printf("er_Manufacturer    =");
        printf("%d,",cd->cd_Rom.er_Manufacturer);
        printf("$%x,",cd->cd_Rom.er_Manufacturer);
        printf("(~$%4x)\n",~cd->cd_Rom.er_Manufacturer);
```

```
    printf("er_Product          =");
        printf("%d,",cd->cd_Rom.er_Product);
        printf("$%x,",cd->cd_Rom.er_Product);
        printf("(~$%x)\n",~cd->cd_Rom.er_Product);

    printf("er_Type             =$%x\n",cd->cd_Rom.er_Type);

    printf("er_Flags            =");
        printf("$%x\n",cd->cd_Rom.er_Flags);

    /* Questi valori sono generati quando il software AUTOCONFIG
       riloca la scheda */
    printf("cd_BoardAddr        =$%lx\n",cd->cd_BoardAddr);
    printf("cd_BoardSize        =");
    printf("$%lx (%ldK)\n",cd->cd_BoardSize,(ULONG)(cd->BoardSize)/1024);
    printf("cd_Flags            =$%x\n",cd->cd_Flags);
    }
CloseLibrary(ExpansionBase);
}
```

TASTIERA

Questa appendice contiene le specifiche d'interfacciamento con la tastiera per A1000, A500 e A2000.

La tastiera viene collegata all'Amiga tramite un cavo con quattro connessioni primarie. Si tratta dell'alimentazione a 5 V, del collegamento a terra, e di due segnali chiamati KCLK (clock della tastiera) e KDAT (dati della tastiera). KCLK è un segnale unidirezionale controllato dalla tastiera stessa; KDAT viene controllato sia dalla tastiera sia dal computer. Entrambi i segnali sono a collettore aperto: vi sono resistenze di pull-up sia nella tastiera (all'interno del suo microprocessore) sia nel computer.

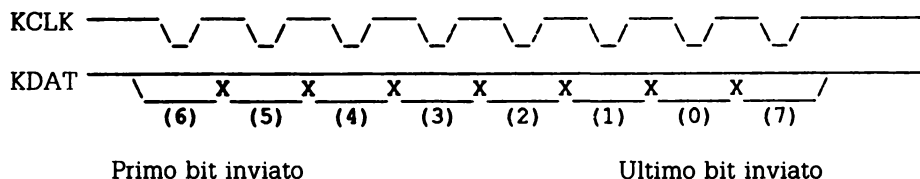
Comunicazioni con la tastiera

La tastiera trasmette serialmente all'unità centrale dati formati ciascuno da 8 bit. Prima che cominci la trasmissione, KCLK e KDAT sono entrambi alti. Poi la tastiera imposta KDAT a seconda del primo bit da inviare e manda quindi un impulso su KCLK (prima basso e poi alto). Ciò viene ripetuto fino a che non sono stati trasmessi tutti gli otto bit di dati. Al termine dell'ultimo impulso di KCLK, KDAT ritorna nuovamente alto.

Quando il computer ha ricevuto l'ottavo bit, deve abbassare la linea KDAT per almeno 1 microsecondo, per confermare alla tastiera la ricezione dei dati. *La tastiera dev'essere capace di reagire a impulsi superiori o uguali a 1 microsecondo.* Il software deve comunque tenere bassa la linea KDAT per almeno 85 microsecondi, per assicurare la compatibilità con tutti i modelli di tastiera.

Tutti i codici trasmessi sono ruotati verso sinistra di un bit prima del loro invio. L'ordine di trasmissione dei bit è pertanto 6-5-4-3-2-1-0-7 Ciò è dovuto alla necessità di trasmettere per ultimo il flag di pressione/rilascio del tasto, in modo da causare la trasmissione di un evento di rilascio nel caso in cui la tastiera debba rimediare a una condizione di perdita di sincronia (se ne parla dettagliatamente in seguito).

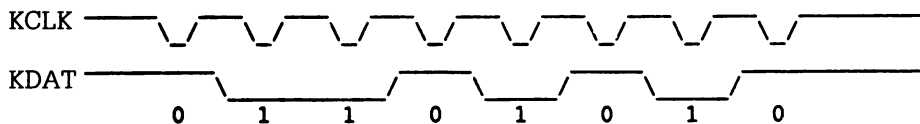
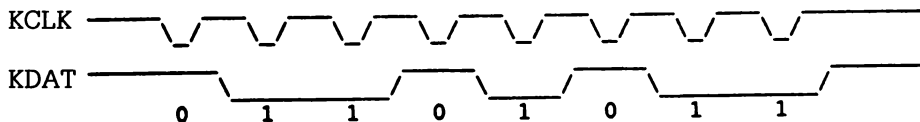
Il segnale KDAT è attivo basso: un livello alto (+5V) viene cioè interpretato come 0, e un livello

basso (0V) come 1.

Il microprocessore della tastiera imposta il segnale KDAT circa 20 microsecondi prima di abbassare KCLK. Questo rimane basso per altri 20 microsecondi circa e quindi ritorna alto. La tastiera attende ancora altri 20 microsecondi prima di modificare il segnale KDAT.

Pertanto, il tempo di di trasmissione è di circa 60 microsecondi per bit, che corrisponde a una velocità di 17 Kbit/sec.

Codici dei tasti

Ogni tasto è associato a un valore espresso sempre su 7 bit (si veda la tavola al termine del capitolo). L'ottavo bit viene usato come flag per indicare la condizione di pressione o rilascio; uno 0 (livello alto) indica che il tasto è stato premuto, mentre un 1 (livello basso) indica che il tasto è stato rilasciato (il tasto CAPS LOCK, che agisce in modo diverso, viene discusso a parte).

Come esempio, ecco un diagramma della trasmissione relativa alla pressione del tasto "B". Il codice corrispondente è \$35 = %00110101; a causa della rotazione descritta, i bit vengono trasmessi nell'ordine 01101010.

Nell'esempio successivo, la "B" viene rilasciata. Il codice è sempre \$35, eccetto che il bit 7 viene impostato a 1 per indicare che si tratta di un evento di rilascio, il che dà origine al codice \$B5 = %10110101 e quindi, dopo la rotazione, %01101011.

Tasto CAPS LOCK

Questo tasto si differenzia dagli altri perché genera la trasmissione di un codice soltanto quando viene premuto, mai quando viene rilasciato. Il bit di pressione/rilascio, tuttavia, viene sempre utilizzato. Quando la pressione del tasto CAPS LOCK provoca l'accensione del led, questo bit vale 0. Vale 1 quando la pressione del tasto ne provoca invece lo spegnimento.

Perdita della sincronia

Disturbi elettrici o altri eventi possono provocare la perdita della sincronia tra la tastiera e il computer. In parole povere, significa che la tastiera ha terminato la trasmissione di un codice ma il computer, non avendolo ricevuto correttamente, è ancora in attesa di dati.

Quando si verifica questa eventualità, la tastiera non riceve ovviamente l'impulso da parte del computer che indica la corretta ricezione dei dati. Se questo impulso non arriva entro 143 ms dall'ultimo impulso di clock generato dalla tastiera, questa deve ritenere che l'elaboratore sia ancora in attesa del resto della trasmissione, e che quindi sia stata perduta la sincronia. È compito quindi della tastiera ristabilire questa sincronia, entrando in "resync mode" In questo modo operativo, la tastiera invia un 1 e attende il segnale di ricevuto. Se questo non arriva entro 143 ms, invia un altro 1 e si rimette in attesa. Questo ciclo continua fino a che non giunge il segnale di dati ricevuti.

Una volta che la sincronia sia stata ripristinata, il codice ricevuto dall'elaboratore avrà ovviamente perso ogni significato. Ed ecco il motivo per cui il flag di pressione/rilascio viene trasmesso per ultimo: dal momento che la tastiera per ripristinare la sincronia invia una serie di valori 1, il codice ricevuto apparirà al computer come il rilascio di un tasto, cosa sicuramente meno pericolosa della sua pressione.

Ogni volta che si verifica questa perdita di sincronia, la tastiera deve ritenere che l'ultimo codice trasmesso non sia stato ricevuto correttamente. Dal momento che il computer non ha modo di sapere che vi è stata una perdita di sincronia, diventa una responsabilità della tastiera informare l'unità centrale. A questo scopo invia al computer lo speciale codice $F9, "perdita di sincronia"; e ritrasmette quindi il codice che non è stato ricevuto correttamente.

La sola ragione per cui occorre agire in questo modo è per avvertire il software che qualcosa potrebbe essere andato storto. Il codice $F9, infatti, non aiuta il processo di recupero, poiché il tasto errato è già stato trasmesso e non può essere cancellato, e il codice corretto avrebbe potuto essere semplicemente ritrasmesso, senza avvertire il computer che vi era qualcosa di errato in quello precedente.

Sequenza d'inizializzazione

Vi sono due circostanze che mettono in funzione la tastiera: il computer che viene acceso con la tastiera già collegata, oppure la tastiera che viene collegata a una macchina già accesa. Il computer e la tastiera devono essere in grado di gestire ognuno dei due casi senza causare problemi.

La prima cosa che la tastiera fa all'accensione è un processo di autodiagnosi. Ciò comprende un checksum della ROM, un semplice test della RAM e un test del watchdog timer (letteralmente "timer cane da guardia"). Ogniqualvolta la tastiera viene avviata, o dopo un reset, non deve trasmettere alcun codice fino a che non è entrata in sincronia con il computer. La procedura è quella descritta: il lento invio di segnali 1 finché non giunge il segnale di ricevuto.

Se la tastiera viene collegata prima dell'accensione dell'elaboratore, passano alcuni secondi durante i quali il computer esegue i suoi controlli interni e il processo di boot. La tastiera deve comunque continuare a inviare valori 1 finché non riceve il segnale di risposta.

Se la tastiera viene collegata a un computer già acceso, non dovrebbero servire più di otto trasmissioni di dati per raggiungere la sincronia. L'elaboratore, però, in qualunque stato si trovi, non dev'essere disturbato dalla "spazzatura" ricevuta inizialmente. Anche in questo caso, ricevendo comunque un codice corrispondente al rilascio di un tasto, il danno dovrebbe essere minimo. Il driver della tastiera deve prevedere una simile eventualità così come devono farlo tutti i programmi che utilizzano direttamente i codici inviati.

La tastiera non deve trasmettere il codice $F9 di perdita della sincronia in questi casi, ma solo in seguito a un processo di risincronizzazione dovuto alla mancata ricezione del segnale di ricevuto, come descritto nel paragrafo precedente.

Una volta che computer e tastiera sono in sincronia, la tastiera deve comunicare al sistema il risultato del test di autodiagnosi. Se è fallito, deve trasmettere il codice $FC (e in tal caso la tastiera non attende il segnale di ricevuto). Subito dopo, la tastiera entra in un loop in cui fa lampeggiare il led del tasto CAPS LOCK in modo da far sapere all'utente qual è il motivo del mancato funzionamento. Si tratta di serie di uno, due, tre o quattro lampeggi, approssimativamente una serie al secondo:

Un lampeggio	Fallimento del checksum della ROM
Due lampeggi	Fallimento del test della RAM
Tre lampeggi	Fallimento del test del watchdog timer
Quattro lampeggi	Esiste un cortocircuito tra due linee della matrice o uno dei sette tasti speciali (non ancora disponibile).

Se il test ha successo, la tastiera procede invece a inviare i codici dei tasti. Innanzitutto trasmette un codice di inizio ($FD) seguito dai codici dei tasti che al momento sono premuti (il flag di pressione/rilascio deve indicare che il tasto è premuto). Quando sono stati trasmessi tutti questi codici (in genere non ce n'è nessuno), viene inviato il codice $FE che indica la fine della sequenza iniziale. Infine, si spegne il led del tasto CAPS LOCK. Ciò indica la fine della sequenza di avvio e l'inizio delle normali operazioni.

Il normale ordine degli eventi è quindi: accensione, sincronizzazione, inizio della sequenza di avvio ($FD), conclusione della sequenza di avvio ($FE).

Avvertimento di reset

(Disponibile solamente su alcune tastiere di A1000 e A2000).

Alla tastiera è affidato anche il compito di provvedere al reset del computer su richiesta dell'utente. Ciò avviene tramite la pressione simultanea del tasto CTRL e dei due tasti AMIGA.

La tastiera risponde a questa condizione terminando l'eventuale trasmissione in corso e inviando quindi un segnale di avvertimento ($78) all'Amiga. Ciò permette al software di concludere operazioni in corso di svolgimento (come operazioni di DMA o altro) e di prepararsi al reset.

Vi è una specifica serie di operazioni da seguire per assicurarsi che l'Amiga sia in condizioni di rispondere all'avvertimento. La tastiera invia infatti due codici di avvertimento. L'Amiga deve rispondere al primo come a un normale codice, altrimenti la tastiera passa direttamente al reset vero e proprio. Al secondo codice di avvertimento l'Amiga deve abbassare il segnale KDAT per almeno 250 millisecondi, altrimenti anche in questo caso la tastiera passa

direttamente all'hard reset. Se entrambe queste condizioni vengono soddisfatte, l'Amiga ha dieci secondi per terminare le operazioni in corso. Quando la linea KDAT viene nuovamente alzata, o comunque dopo dieci secondi, si passa all'hard reset.

Hard reset

(Avviene dopo l'avvertimento di reset. Vale per tutte le tastiere eccetto che per quella dell'A500).

La tastiera provoca il reset dell'Amiga abbassando la linea KCLK e dando il via a un conteggio di 500 millisecondi. Quando uno o più dei tasti sono stati rilasciati *e* sono trascorsi 500 millisecondi, KCLK torna alto. Cinquecento millisecondi è il tempo minimo per il quale KCLK dev'essere tenuto basso. Il tempo massimo dipende invece da quanto a lungo l'utente tiene premuti i tre tasti di reset.

La circuiteria sulla piastra madre dell'Amiga è in grado di registrare un impulso di 500 millisecondi da parte di KCLK.

Dopo il rilascio del segnale KCLK, la tastiera passa direttamente alla sequenza di avvio che la reinizializza in maniera analoga a quanto accade all'accensione. Deve pertanto ritrasmettere, a sincronizzazione avvenuta, la sequenza di avvio.

Codici speciali

Ecco un sommario dei principali codici utilizzati dalla tastiera per la comunicazione con l'unità centrale.

Questi codici sono numeri a 8 bit: non possiedono nessun flag di pressione/rilascio. Tuttavia, l'ordine della trasmissione dei bit è lo stesso illustrato in precedenza.

78	Avvertimento di reset. Sono stati premuti i tasti CTRL-AMIGA-AMIGA.
F9	L'ultimo codice era errato, il prossimo è lo stesso ritrasmesso.
FA	Overflow del buffer interno della tastiera.
FB	Non utilizzato (indicava fallimento del controller).
FC	Autodiagnosi della tastiera dopo un errore.
FD	Inizio della sequenza di avvio.
FE	Termine della sequenza di avvio.
FF	Non utilizzato (indicava interrupt).

Tavola della matrice della tastiera

Colonna	Riga 5 (Bit 7)	Riga 4 (Bit 6)	Riga 3 (Bit 5)	Riga 2 (Bit 4)	Riga 1 (Bit 3)	Riga 0 (Bit 2)
15 (PD.7)	vuoto 0E	vuoto 1C	vuoto 2C	vuoto 47	vuoto 48	vuoto 49
14 (PD.6)	* nota 1 5D	SHIFT nota 2 30	CAPS LOCK 62	TAB 42	~ ` 00	ESC 45
13 (PD.5)	+ nota 1 5E	Z 31	A 20	Q 10	! 1 01	(nota 1 5A
12 (PD.4)	9 nota 3 3F	X 32	S 21	W 11	@ 2 02	F1 50
11 (PD.3)	6 nota 3 2F	C 33	D 22	E 12	# 3 03	F2 51
10 (PD.2)	3 nota 3 1F	V 34	F 23	R 13	$ 4 04	F3 52
9 (PD.1)	. nota 3 3C	B 35	G 24	T 14	% 5 05	F4 53
8 (PD.0)	8 nota 3 3E	N 36	H 25	Y 15	^ 6 06	F5 54
7 (PC.7)	5 nota 3 2E	M 37	J 26	U 16	& 7 07	) nota 1 5B
6 (PC.6)	2 nota 3 1E	< , 38	K 27	I 17	* 8 08	F6 55
5 (PC.5)	ENTER nota 3 43	> . 39	L 28	O 18	(9 09	/ nota 1 5C

4 (PC.4)	7 nota 3 3D	? / 3A	: ; 29	P 19	) 0 0A	F7 56
3 (PC.3)	4 nota 3 2D	vuoto 3B	" ' 2A	{ [1A	_ - 0B	F8 57
2 (PC.2)	1 nota 3 1D	SPAZIO 40	RET nota 2 2B	}] 1B	+ = 0C	F9 58
1 (PC.1)	0 nota 3 0F	BACK SPACE 41	DEL 46	RETURN 44	| \ 0D	F10 59
0 (PC.0)	- nota 3 4A	CURS. GIÙ 4D	CURS. DESTRA 4E	CURS. SINISTRA 4F	CURS. SU 4C	HELP 5F

nota 1: Solo A500 e A2000 (tastierina numerica)
nota 2: Solo su tastiere internazionali. Il tasto di codice $30 viene ricavato dallo Shift sinistro. Il tasto di codice $2B dal Return.
nota 3: Tastierina numerica.

I tasti seguenti non fanno parte della matrice e pertanto non generano mai caratteri "fantasma".

Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AMIGA SINISTRO 66	ALT SINISTRO 64	SHIFT SINISTRO 60	CTRL 63	AMIGA DESTRO 67	ALT DESTRO 65	SHIFT DESTRO 61

Note generali

Il connettore a 23 pin femmina sul retro dell'unità centrale viene usato come interfaccia per il controllo di dispositivi in grado di generare e ricevere dati di tipo MFM. Si può accedere a questa interfaccia direttamente come risorsa, oppure tramite il controllo di un driver. Le pagine seguenti trattano entrambi i casi.

Sommario

PIN	NOME	CLASSE	NOTE
1	RDY*	I/O	Segnali d'identificazione e di pronto
2	DKRD*	I	Input di dati MFM
3	GRND	G	Terra
4	GRND	G	Terra
5	GRND	G	Terra
6	GRND	G	Terra
7	GRND	G	Terra
8	MTRXD*	O	Controllo del motore
9	SEL2B*	O*	Selezione disk drive 2
10	DRESB*	O	Reset
11	CHNG*	I/O	Disco estratto dal disk drive
12	+5V	PWR	540 mA medi, 870 mA sorgenti
13	SIDEB*	O	Selezione lato del disco
14	WPRO*	I/O	Protezione da scrittura
15	TK0*	I/O	Traccia 0
16	DKWEB*	O	Abilitazione della scrittura

17	DKWDB*	O	Dati MFM in output
18	STEPB*	O	Step della testina
19	DIRB	O	Direzione di movimento della testina
20	SEL3B*	O*	Selezione disk drive 3
21	SEL1B*	O*	Selezione disk drive 1
22	INDEX*	I/O	Index
23	+12V	PWR	120 mA medi, 370 mA sorgenti

LEGENDA DELLE CLASSI:

G Terra

I Input portato a 5V da 1 KΩ

I/O Input per il driver, ma bidirezionale (1KΩ di pull-up).

O Output portato a 5V da 1KΩ

O* Output, separa le risorse.

PWR Disponibile per uso esterno, usato per l'alimentazione del drive esterno.

Segnali di controllo del disco

L'elenco che segue descrive l'interfaccia sotto il controllo del driver.

SEL1B*, SEL2B*, SEL3B*
Linee di selezione per i tre drive esterni, attive basse.

TK0* Il drive selezionato porta questo segnale basso quando la sua testina è sulla traccia 0.

RDY* Quando il segnale del motore è attivo, questa linea indica che il drive selezionato è installato e sta ruotando alla velocità corretta. Il driver ignora il segnale. Quando il segnale del motore non è attivo, questa linea viene usata per leggere informazioni d'identificazione del drive. Vedere più avanti la descrizione completa.

WPRO* (Pin 14)
Il drive selezionato attiva questo segnale quando contiene un disco protetto da scrittura.

INDEX* (Pin 22)
Il drive selezionato invia un impulso su questa linea a ogni rivoluzione del disco.

SIDEB* (Pin 13)
Il sistema invia questo segnale a tutti i drive: basso per il lato 1, alto per il lato 0.

STEPB* (Pin 18)
Impulso che causa lo spostamento della testina del drive selezionato.

DIRB (Pin 19)
Il sistema invia questo segnale alto o basso per indicare al drive selezionato la direzione in cui si deve spostare la testina all'arrivo dell'impulso STEPB*. Se il segnale è basso, lo spostamento avviene verso l'interno del disco (tracce con numero maggiore); se il segnale è basso, lo spostamento avviene verso l'esterno.

DKRD* (Pin 2)
Il drive selezionato invia su questa linea i dati letti.

DKWDB*(Pin 17)
Il drive invia i dati in scrittura al disco tramite questo segnale. I dati vengono scritti soltanto se anche il segnale DKWEB* è attivo (basso). I dati vengono scritti solo sui drive selezionati.

DKWEB*(Pin 16)
Questo segnale causa la scrittura su disco dei dati inviati tramite il segnale DKWDB* da parte del drive selezionato.

CHNG* (Pin 11)
Il drive selezionato provoca l'abbassamento di questo segnale ogni volta che viene impostato il suo latch interno di "disco cambiato". L'impostazione avviene al momento dell'accensione del drive oppure quando viene rimosso un disco. Per provocare l'aggiornamento del segnale, il sistema deve selezionare il drive e tentare lo spostamento della testina. Ovviamente, se non vi è alcun disco inserito, il segnale rimane immutato.

MTRXD*(Pin 8)
Questo segnale controlla il motore di tutti e quattro i drive. Quando il sistema vuole avviare il motore di un drive, prima lo deseleziona, poi abbassa il segnale MTRXD* e infine lo seleziona. Per fermarlo, invece, deve deselezionare il drive, riportare alto MTRXD*, e selezionare il drive nuovamente. Il sistema imposta MTRXD* almeno 1,4 microsecondi prima di selezionare il drive, e non lo modifica per un ulteriore periodo di almeno 1,4 microsecondi. Tutti i drive esterni devono possedere una logica equivalente a un flip-flop di tipo D, in cui l'input D è rappresentato dal segnale MTRXD*, e il cui clock è attivato dalla transizione alto→basso di SELxB*. Come accennato, sia il tempo di setup sia quello di hold di MTRXD* rispetto a SELxB*, devono essere di almeno 1,4 microsecondi. L'output di questo flip-flop controlla il motore del drive. Pertanto, il sistema può controllare tutti e quattro i motori tramite un unico segnale, MTRXD*.

DRESB* (Pin 10)
Questo segnale è collegato al segnale di reset del sistema. Tre cose possono farlo diventare attivo (basso):

- L'accensione del sistema (DRESB* viene attivato per circa un secondo).
- La CPU che esegue un'istruzione RESET (attivato per circa 17 microsecondi).
- Hard reset proveniente dalla tastiera (rimane attivo fino a che sono premuti i tasti di reset).

I drive esterni dovrebbero rispondere a questo segnale disattivando il flip-flop del loro motore e proteggendosi da scrittura.

Questa stessa procedura dev'essere eseguita dai drive esterni quando la tensione sulla linea dei +5 V scende a 3,75 V o meno.

Identificazione del dispositivo

Questa interfaccia prevede un metodo per stabilire il tipo dei drive a essa collegati. La sequenza delle operazioni è la seguente:

1. Abbassare il segnale MTRXD*.
2. Abbassare il segnale SELxB*.
3. Alzare il segnale SELxB*.
4. Alzare il segnale MTRXD*.
5. Abbassare il segnale SELxB*.
6. Alzare il segnale SELxB*.
7. Abbassare il segnale SELxB*.
8. Leggere e salvare lo stato di RDY*.
9. Alzare il segnale SELxB*.

Ripetere per 15 volte ancora i passi da 6 a 9.

Si convertono quindi i 16 valori letti da RDY* in una word da 16 bit. Il primo valore letto costituisce il bit più significativo e così via. Il valore ottenuto rappresenta l'identificatore del dispositivo.

Sono stati definiti i seguenti valori:

```
0000 0000 0000 0000     Riservato
1111 1111 1111 1111     Amiga standard 3,5
1010 1010 1010 1010     Riservato
0101 0101 0101 0101     48 TPI, doppia faccia, doppia densita'
1000 0000 0000 0000     Riservato
0111 1111 1111 1111     Riservato
0000 1111 xxxx xxxx     Disponibili
1111 0000 xxxx xxxx     Riservati
xxxx 0000 0000 0000     Riservati
xxxx 1111 1111 1111     Riservati
0011 0011 0011 0011     Riservato
1100 1100 1100 1100     Riservato
```

J FILE INCLUDE PER GLI ESEMPI HARDWARE

Questa appendice contiene un file include che ridefinisce i nomi dei registri definiti nelle appendici A e B in modo che possano esssere utilizzati con i file include di sistema. L'uso di questi nomi permette infatti, negli esempi di questo manuale, di porre in risalto ciò che il codice sta realmente facendo, senza bisogno di sforzare la memoria con nomi astrusi.

Tutti gli esempi di questo manuale utilizzano i nomi definiti in questo file.

```
        IFND    HARDWARE_HW_EXAMPLES_I
HARDWARE_HW_EXAMPLES_I  SET     1

**
**      Filename: hardware/hw_examples.i
**      Release: 1.3
**
**
**
**      (C) Copyright 1985,1986,1987,1988,1989 Commodore-Amiga, Inc.
**          All Rights Reserved
**
************************************************************************

        IFND    HARDWARE_CUSTOM_I
        INCLUDE "hardware/custom.i"
        ENDC

************************************************************************
*
*  Questo file e' progettato per essere usato insieme agli
*  esempi proposti nel corso del volume. Vi vengono definiti i
```

```
*  nomi dei registri sulla base di quelli contenuti nel file
*  hardware/custom.i.
*
***********************************************************************
*
*  Questa istruzione del Copper lo fa entrare in attesa per
*  sempre, dal momento che la condizione specificata non si
*  puo' mai verificare.
*
COPPER_HALT     equu    $FFFFFFFE
*
***********************************************************************
*
*  Il seguente e' l'indirizzo base dei chip custom. Equivale
*  a _custom eseguendo il link con amiga.lib.
*
CUSTOM          equ     $DFF000
*
*  Vari registri di controllo
*
DMACONR         equ     dmaconr
VPOSR           equ     vposr
VHPOSR          equ     vhposr
JOY0DAT         equ     joy0dat
JOY1DAT         equ     joy1dat
CLXDAT          equ     clxdat
CLXCON          equ     clxcon
ADKCON          equ     adkcon
ADKCONR         equ     adkconr
POT0DAT         equ     pot0dat
POT1DAT         equ     pot1dat
POTINP          equ     potinp
SERDATR         equ     serdatr
INTENAR         equ     intenar
INTREQR         equ     intreqr
REFPTR          equ     refptr
VPOSW           equ     vposw
VHPOSW          equ     vhposw
SERDAT          equ     serdat
SERPER          equ     serper
POTGO           equ     potgo
JOYTEST         equ     joytest
STREQU          equ     strequ
STRVBL          equ     strvbl
STRHOR          equ     strhor
STRLONG         equ     strlong
DIWSTRT         equ     diwstrt
DIWSTOP         equ     diwstop
DDFSTRT         equ     ddfstrt
DDFSTOP         equ     ddfstop
DMACON          equ     dmacon
```

```
INTENA          equ     intena
INTREQ          equ     intreq
*
*  Registri di controllo del disco
*
DSKBYTR         equ     dskbytr
DSKPT           equ     dskpt
DSKPTH          equ     dskpt
DSKPTL          equ     dskpt+$02
DSKLEN          equ     dsklen
DSKDAT          equ     dskdat
DSKDATR         equ     dskdatr
DSKSYNC         equ     dsksync
*
*  Registri del Blitter
*
BLTCON0         equ     bltcon0
BLTCON1         equ     bltcon1
BLTAFWM         equ     bltafwm
BLTALWM         equ     bltalwm
BLTCPT          equ     bltcpt
BLTCPTH         equ     bltcpt
BLTCPTL         equ     bltcpt+$02
BLTBPT          equ     bltbpt
BLTBPTH         equ     bltbpt
BLTBPTL         equ     bltbpt+$02
BLTAPT          equ     bltapt
BLTAPTH         equ     bltapt
BLTAPTL         equ     bltapt+$02
BLTDPT          equ     bltdpt
BLTDPTH         equ     bltdpt
BLTDPTL         equ     bltdpt+$02
BLTSIZE         equ     bltsize
BLTCMOD         equ     bltcmod
BLTBMOD         equ     bltbmod
BLTAMOD         equ     bltamod
BLTDMOD         equ     bltdmod
BLTCDAT         equ     bltcdat
BLTBDAT         equ     bltbdat
BLTADAT         equ     bltadat
BLTDDAT         equ     bltddat
*
*  Registri di controllo del Copper
*
COPCON          equ     copcon
COPINS          equ     copins
COPJMP1         equ     copjmp1
COPJMP2         equ     copjmp2
COP1LC          equ     cop1lc
COP1LCH         equ     cop1lc
COP1LCL         equ     cop1lc+$02
```

```
COP2LC          equ     cop2lc
COP2LCH         equ     cop2lc
COP2LCL         equ     cop2lc+$02
*
*  Registri dei canali audio
*
AUD0LC          equ     aud0
AUD0LCH         equ     aud0
AUD0LCL         equ     aud0+$02
AUD0LEN         equ     aud0+$04
AUD0PER         equ     aud0+$06
AUD0VOL         equ     aud0+$08
AUD0DAT         equ     aud0+$0A

AUD1LC          equ     aud1
AUD1LCH         equ     aud1
AUD1LCL         equ     aud1+$02
AUD1LEN         equ     aud1+$04
AUD1PER         equ     aud1+$06
AUD1VOL         equ     aud1+$08
AUD1DAT         equ     aud1+$0A

AUD2LC          equ     aud2
AUD2LCH         equ     aud2
AUD2LCL         equ     aud2+$02
AUD2LEN         equ     aud2+$04
AUD2PER         equ     aud2+$06
AUD2VOL         equ     aud2+$08
AUD2DAT         equ     aud2+$0A

AUD3LC          equ     aud3
AUD3LCH         equ     aud3
AUD3LCL         equ     aud3+$02
AUD3LEN         equ     aud3+$04
AUD3PER         equ     aud3+$06
AUD3VOL         equ     aud3+$08
AUD3DAT         equ     aud3+$0A
*
*  Registri di controllo dei bitplane
*
BPL1PT          equ     bplpt+$00
BPL1PTH         equ     bplpt+$00
BPL1PTL         equ     bplpt+$02
BPL2PT          equ     bplpt+$04
BPL2PTH         equ     bplpt+$04
BPL2PTL         equ     bplpt+$06
BPL3PT          equ     bplpt+$08
BPL3PTH         equ     bplpt+$08
BPL3PTL         equ     bplpt+$0A
BPL4PT          equ     bplpt+$0C
BPL4PTH         equ     bplpt+$0C
```

```
BPL4PTL         equ     bplpt+$0E
BPL5PT          equ     bplpt+$10
BPL5PTH         equ     bplpt+$10
BPL5PTL         equ     bplpt+$12
BPL6PT          equ     bplpt+$14
BPL6PTH         equ     bplpt+$14
BPL6PTL         equ     bplpt+$16

BPLCON0         equ     bplcon0
BPLCON1         equ     bplcon1
BPLCON2         equ     bplcon2
BPL1MOD         equ     bpl1mod
BPL2MOD         equ     bpl2mod

BPL1DATA        equ     bpldat+$00
BPL2DATA        equ     bpldat+$02
BPL3DATA        equ     bpldat+$04
BPL4DATA        equ     bpldat+$06
BPL5DATA        equ     bpldat+$08
BPL6DATA        equ     bpldat+$0A
*
*  Registri di controllo degli sprite
*
SPR0PT          equ     sprpt+$00
SPR0PTH         equ     sprpt+$00
SPR0PTL         equ     sprpt+$02
SPR1PT          equ     sprpt+$04
SPR1PTH         equ     sprpt+$04
SPR1PTL         equ     sprpt+$06
SPR2PT          equ     sprpt+$08
SPR2PTH         equ     sprpt+$08
SPR2PTL         equ     sprpt+$0A
SPR3PT          equ     sprpt+$0C
SPR3PTH         equ     sprpt+$0C
SPR3PTL         equ     sprpt+$0E
SPR4PT          equ     sprpt+$10
SPR4PTH         equ     sprpt+$10
SPR4PTL         equ     sprpt+$12
SPR5PT          equ     sprpt+$14
SPR5PTH         equ     sprpt+$14
SPR5PTL         equ     sprpt+$16
SPR6PT          equ     sprpt+$18
SPR6PTH         equ     sprpt+$18
SPR6PTL         equ     sprpt+$1A
SPR7PT          equ     sprpt+$1C
SPR7PTH         equ     sprpt+$1C
SPR7PTL         equ     sprpt+$1E

SPR0POS         equ     spr+$00
SPR0CTL         equ     SPR0POS+$02
SPR0DATA        equ     SPR0POS+$04
```

```
SPR0DATB        equ     SPR0POS+$06

SPR1POS         equ     spr+$08
SPR1CTL         equ     SPR1POS+$02
SPR1DATA        equ     SPR1POS+$04
SPR1DATB        equ     SPR1POS+$06

SPR2POS         equ     spr+$10
SPR2CTL         equ     SPR2POS+$02
SPR2DATA        equ     SPR2POS+$04
SPR2DATB        equ     SPR2POS+$06

SPR3POS         equ     spr+$18
SPR3CTL         equ     SPR3POS+$02
SPR3DATA        equ     SPR3POS+$04
SPR3DATB        equ     SPR3POS+$06

SPR4POS         equ     spr+$20
SPR4CTL         equ     SPR4POS+$02
SPR4DATA        equ     SPR4POS+$04
SPR4DATB        equ     SPR4POS+$06

SPR5POS         equ     spr+$28
SPR5CTL         equ     SPR5POS+$02
SPR5DATA        equ     SPR5POS+$04
SPR5DATB        equ     SPR5POS+$06

SPR6POS         equ     spr+$30
SPR6CTL         equ     SPR6POS+$02
SPR6DATA        equ     SPR6POS+$04
SPR6DATB        equ     SPR6POS+$06

SPR7POS         equ     spr+$38
SPR7CTL         equ     SPR7POS+$02
SPR7DATA        equ     SPR7POS+$04
SPR7DATB        equ     SPR7POS+$06
*
*  Registri di colore
*
COLOR00         equ     color+$00
COLOR01         equ     color+$02
COLOR02         equ     color+$04
COLOR03         equ     color+$06
COLOR04         equ     color+$08
COLOR05         equ     color+$0A
COLOR06         equ     color+$0C
COLOR07         equ     color+$0E
COLOR08         equ     color+$10
COLOR09         equ     color+$12
COLOR10         equ     color+$14
COLOR11         equ     color+$16
```

```
COLOR12         equ     color+$18
COLOR13         equ     color+$1A
COLOR14         equ     color+$1C
COLOR15         equ     color+$1E
COLOR16         equ     color+$20
COLOR17         equ     color+$22
COLOR18         equ     color+$24
COLOR19         equ     color+$26
COLOR20         equ     color+$28
COLOR21         equ     color+$2A
COLOR22         equ     color+$2C
COLOR23         equ     color+$2E
COLOR24         equ     color+$30
COLOR25         equ     color+$32
COLOR26         equ     color+$34
COLOR27         equ     color+$36
COLOR28         equ     color+$38
COLOR29         equ     color+$3A
COLOR30         equ     color+$3C
COLOR31         equ     color+$3E

                ENDC
```

GLOSSARIO

ALTA RISOLUZIONE
Modo video tramite il quale vengono visualizzati 640 pixel per ogni linea orizzontale.

AMIGADOS
Il sistema operativo dell'Amiga.

AMPIEZZA
L'ampiezza della forma d'onda. Influisce sul volume del suono.

ANIMAZIONE DI PLAYFIELD
Un metodo di animazione basato sullo spostamento tramite il Blitter di aree appartenenti ai playfield.

BARREL SHIFTER
Un particolare circuito, contenuto nel Blitter, che permette lo scorrimento simultaneo di un'intera riga di dati.

BASSA RISOLUZIONE
Modo video che permette la visualizzazione di 320 pixel per ogni linea orizzontale.

BITMAP
La completa definizione di un'immagine video in memoria. È composta da uno o più bitplane e dalle informazioni sulle loro dimensioni.

BITPLANE
Una serie di word contigue organizzate come un'area rettangolare.

BLITTER
Il canale DMA usato per copiare aree di memoria e tracciare linee.

CLI
L'interfaccia linea comando utilizzata dal sistema.

COLLISIONE
Si verifica una collisione quando sprite, playfield o altri oggetti video si sovrappongono nella medesima posizione di schermo od oltrepassano limiti predefiniti.

CONTROLLER
Dispositivo hardware, come un mouse o una penna ottica, usato per muovere un puntatore o comunque fornire un input al sistema.

CONVERTITORE DIGITALE-ANALOGICO
Un circuito che converte numeri binari in livelli analogici.

COORDINATE
Una coppia di numeri nella forma (x,y), dove x rappresenta l'offset dal lato sinistro del video e y quello dal lato superiore.

COPPER
Coprocessore sincronizzato con il pennello elettronico; viene impiegato per la gestione dell'immagine video.

COPROCESSORE
Processore che aggiunge un proprio set di istruzioni a quello del processore principale.

CPU
Central Processing Unit, unità di calcolo centrale. Il microprocessore principale.

DISPLAY
Tutto ciò che è visibile sullo schermo del monitor o del televisore collegato all'Amiga.

DISTORSIONE DI ALIASING
Un effetto collaterale del campionamento audio che causa la comparsa di due frequenze addizionali, distorcendo l'output sonoro.

DMA
Direct Memory Access, accesso diretto alla memoria. Una tecnica grazie alla quale dispositivi intelligenti possono leggere e scrivere direttamente la memoria senza dover interrompere la CPU.

EXEC
La serie di funzioni a basso livello che controlla il sistema multitasking dell'Amiga.

FINESTRA VIDEO
La parte di bitmap selezionata per la visualizzazione. Indica anche le dimensioni dell'area visibile sullo schermo.

FREQUENZA
Numero che indica quante volte al secondo si ripete una forma d'onda.

GENLOCK
Interfaccia che consente di utilizzare immagini provenienti da sorgenti esterne insieme a quelle generate dall'Amiga.

HAM
Hold-and-modify, modo video tramite il quale sono visualizzabili contemporaneamente fino a 4096 colori.

INDIREZIONE DEL COLORE
Il metodo usato dall'Amiga per colorare i singoli pixel, nel quale il numero binario formato dai bit associati a un dato pixel individua uno dei 32 registri di colore.

INTERVALLO DI VERTICAL BLANKING:
Il periodo di tempo in cui il pennello elettronico si trova al di fuori del display video.

MEMORIA CONDIVISA
La RAM utilizzata dall'Amiga sia per il video che per l'esecuzione dei programmi.

MINTERM
Una delle otto possibili combinazioni logiche dei bit di dati provenienti dalle tre sorgenti del Blitter.

MODO AUTOMATICO
Nell'uso degli sprite, si ha quando un canale DMA procede automaticamente al trasferimento dati per ogni linea dello sprite. Nella sintesi audio, si riferisce al modo in cui i dati di campionamento vengono letti automaticamente tramite DMA.

MODO COLLEGATO
Nell'uso degli sprite, si ha quando un singolo sprite utilizza due canali DMA per ottenere colori addizionali. Nella generazione di suoni, quando due o più canali sono collegati al fine di ottenere particolari effetti sonori.

MODO DUAL-PLAYFIELD
Un modo video che consente di controllare separatamente due playfield sovrapposti.

MODO INTERLACE
Modo video tramite il quale possono essere visualizzate il doppio delle linee presenti in un normale display.

MODO MANUALE
Nell'uso degli sprite indica la situazione in cui ogni linea dello sprite viene scritta separatamente. Nelle operazioni audio indica il modo in cui le word di dati vengono inviate una alla volta al convertitore digitale-analogico.

MODO NON INTERALACE
Modo video tramite il quale possono essere visualizzate 256 linee sullo schermo.

MODO VIDEO
Uno dei tipi di display selezionabili. Per esempio alta o bassa risoluzione, modo interlace o non interlace, playfield singolo o dual-playfield.

MODULAZIONE DI AMPIEZZA
Un metodo per modificare la qualità di un suono utilizzando un canale audio per alterare l'ampiezza di un altro. Influisce sul volume del suono generato.

MODULAZIONE DI FREQUENZA
Un metodo per modificare la qualità di un suono utilizzando un canale audio per alterare la frequenza di un altro. Influisce sull'altezza del suono generato.

MODULO
Rappresenta il numero di byte da aggiungere ai puntatori al termine di una linea per portarsi all'inizio della successiva.

MULTITASKING
Un sistema tramite il quale diversi programmi possono operare contemporaneamente, senza tener conto della presenza degli altri.

NTSC
Standard televisivo americano, simile, ma parzialmente incompatibile, con lo standard PAL.

OVERSCAN
Area percorsa dal pennello elettronico ma non completamente visibile sullo schermo.

PADDLE
Tipo di controller che utilizza un potenziometro per determinare la posizione degli oggetti sullo schermo.

PAL
Phase Alternate Line, standard televisivo europeo.

PALETTE
La serie dei 32 registri di colore.

PENNA OTTICA
Dispositivo di controllo tramite il quale è possibile agire direttamente sul punto desiderato dello schermo.

PERIODO DI CAMPIONAMENTO
L'intervallo, misurato in cicli di clock, che deve trascorrere tra un campione e il successivo.

PIXEL
Picture Element, il più piccolo elemento singolo indirizzabile sullo schermo.

PLAYFIELD
Uno degli elementi base della grafica dell'Amiga. Lo sfondo sul quale si muovono tutti gli altri elementi del display.

PORTA PARALLELA
Connettore usato per il collegamento di stampanti parallele o altri dispositivi.

PORTA SERIALE
Connettore usato per il collegamento di modem o altri dispositivi seriali.

POTENZIOMETRO
Un dispositivo analogico in grado di controllare una quantità variabile di segnale.

PRIMITIVE
Le funzioni di libreria dell'Amiga che controllano testo, grafica e animazione.

PRIORITÀ VIDEO
Stabilisce l'ordine in cui compaiono sullo schermo i vari oggetti video (playfield e sprite). Oggetti con priorità più bassa sono coperti da quelli con priorità più alta.

PROFONDITÀ
Il numero di bitplane di un playfield.

QUADRO VIDEO
Un completo passaggio dall'alto al basso del pennello elettronico sul video.

RAM
Random Access Memory, memoria ad accesso casuale (volatile).

RASTER
L'area di memoria che costituisce un bitplane.

REGISTRO DI COLORE
Uno dei 32 registri usati nella selezione dei colori.

REGISTRO PUNTATORE
Registro che viene incrementato per puntare a una serie di locazioni di memoria.

REGISTRO STROBE
Registro il cui indirizzo, inviato sul bus, causa il verificarsi di un evento. I dati letti o scritti sono ignorati.

RISOLUZIONE
Il numero di pixel orizzontali e verticali visualizzabili sullo schermo.

RITARDO
Nello scroll orizzontale dei playfield, determina lo spostamento orizzontale della finestra video, controllando la velocità dello scroll.

ROM
Read Only Memory, memoria a sola lettura (permanente).

RUMORE DI QUANTIZZAZIONE
Rumore causato da errori di arrotondamento nella riproduzione di una forma d'onda.

SCALA TEMPERATA
Una scala musicale in cui la frequenza di ogni nota è uguale a quella della precedente moltiplicata per la radice dodicesima di 2.

SCROLL
Movimento orizzontale o verticale di un oggetto video o dello schermo stesso.

SPRITE
Oggetto grafico facilmente spostabile sullo sfondo e da esso indipendente, controllato da uno degli otto canali DMA adibiti a questo scopo.

TASK
Modulo del sistema operativo o programma applicativo.

TASTI ALT
I due tasti immediatamente a destra e a sinistra dei tasti Amiga.

TASTI AMIGA
I due tasti immediatamente a destra e a sinistra della barra spaziatrice.

TRASPARENTE
Una speciale definizione di colore che lascia vedere il colore sottostante. Usato dagli sprite e nel modo dual-playfield.

UART
Il circuito che controlla il collegamento seriale con i dispositivi esterni. È l'acronimo di Universal Asynchronous Receiver/Transmitter: trasmettitore/ricevitore asincrono universale.

VELOCITÀ DI CAMPIONAMENTO
Il numero di campioni inviati in un secondo al convertitore digitale-analogico.

VIDEOCOMPOSITO
Un tipo di segnale video, trasmesso tramite un singolo cavo coassiale, che contiene sia le informazioni riguardanti l'immagine sia quelle riguardanti la sincronia.

INDICE ANALITICO

B

C

D

F

G

H

I

J

L

N

O

P

R

S

T

U

LIBRERIA DI RIFERIMENTO TECNICO DELL'AMIGA

IL MANUALE DELL'HARDWARE DELL'AMIGA

Il volume appartiene alla serie di manuali della libreria di riferimento tecnico per i computer della linea Amiga, realizzati dalla stessa azienda produttrice, la Commodore-Amiga. Si tratta quindi di una fonte d'informazioni ufficiale; uno strumento di riferimento indispensabile per:

- I programmatori in linguaggio Assembly che hanno bisogno d'interagire con la macchina in maniera diretta.
- I progettisti che intendono creare nuove periferiche per l'Amiga.
- Chiunque sia interessato a scoprire come funziona l'hardware dell'Amiga.

Gli argomenti principali di questa approfondita guida, scritta dagli stessi creatori dell'Amiga, sono: l'hardware del Copper, dei playfield, degli sprite, audio, del Blitter, di controllo e d'interfaccia. Non mancano delle utili appendici (registri, mappa di memoria, connettori, interfacciamento) e un glossario.

Alla libreria di riferimento tecnico dell'Amiga appartengono altri due volumi d'informazioni vitali per tutti i programmatori e progettisti dell'Amiga: *Il Manuale del ROM Kernel dell'Amiga: Librerie e Dispositivi*, che contiene spiegazioni dettagliate sull'uso di ogni libreria e di ogni dispositivo dell'Amiga, e *File Include e Autodoc*, che elenca in ordine alfabetico tutti gli autodoc di ogni funzione, dei file include di sistema, e le specifiche di formato dei file IFF.

ISBN 88-7803-018-X
9 788878 030183

Lire 76.000

IL MANUALE DELL'HARDWARE DELL'AMIGA

COMMODORE-AMIGA, INC.

IHT GRUPPO EDITORIALE